

digit **FastTrack**

YOUR HANDY GUIDE TO EVERYDAY TECHNOLOGY

To **Ethical** Hacking

All you ever wanted
to know about ethical
hacking

Introduction to
**Ethical
hacking**

Hacking
the **web/
networks**

Information
Security

Hacking
anything

...and much
more



to

Ethical Hacking

CREDITS

The People Behind This Book

EDITORIAL

Editor	Robert Sovereign-Smith
Head-Copy Desk	Nash David
Writer	Kshitij Sobti

DESIGN AND LAYOUT

Lead Designer	Vijay Padaya
Senior Designer	Baiju NV
Cover Design	Anil T

© 9.9 Mediaworx Pvt. Ltd.

Published by 9.9 Mediaworx

No part of this book may be reproduced, stored in a retrieval system or transmitted in any form or by any means without the prior written permission of the publisher.

June 2010

Free with Digit. Not to be sold separately. If you have paid separately for this book, please email the editor at editor@thinkdigit.com along with details of location of purchase, for appropriate action.

Contents

1 The basics

1.1	What is hacking?	07
1.2	What is ethical hacking?	09

2 Information security

2.1	Passwords	13
2.2	Hashes.....	22

3 Hacking the web / network

3.1	Network hacking.....	37
3.2	Web application hacking.....	54

4 Conclusion

4.1	Hacking... Anything	71
-----	---------------------------	----

5 Appendices

5.1	Before you begin hacking	75
5.2	The Windows registry	77
5.3	Port Lists	80

Introduction

Hacking and ethical hacking are often subject to much misinterpretation. We've tried to deconstruct some of those myths and introduce readers to some of the basic concepts of ethical hacking.

The book itself can be divided into three parts, the Introduction, Information Security, and Hacking the web / network.

In the Introduction to this book, we have tried to give readers a clearer idea of what exactly constitutes hacking. We explore the ethical lines of hacking, and the dissonance between ethical as a legal or moral binding. We question why the term even needs the prefix "ethical". We also take a look at the terms Black Hat hacker and White Hat hacker and how to distinguish between them.

In our second section on Information Security we deal with some of the most basic devices for security and access control: Passwords. In the chapter "Access Denied" we look at exactly what does it take for a password to be secure? We look at what makes a strong password strong and some of the technical limits to cracking password. We also look at brute force and dictionary attacks as means of password cracking.

In the second chapter on "Social Engineering" we explore the social engineering, as a concept of using social means for finding passwords instead of purely electronic means. Here we will look at some of the popular modes of social engineering.

In the chapter "The ethical bit" we explore the ethical uses of knowing how to crack passwords. We see how knowing the processes by which passwords are hacked can help us pick better uncrackable passwords. We look at how one can have a password which is easy to remember and strong at the same time.

In "Hashes" we look at some of the uses of hashes in information security and how they can be cracked to reveal a password. The "What the #!" chapter then deals with what exactly a hash is, how it relates to passwords and how can it be hacked. We explore all these questions and explore the basic function and operation of hashes.

In "Of Rainbows and Salt" we look at hash chains, and rainbow tables,

which are popular means of deciphering hashes. We look at salts, which offer some protection against such means of hacking hashes.


The third section in this Fast Track could actually be looked at as two sections, on “Hacking the network” and “Hacking the web”. It is as such divided into two parts. Hacking over the network, and hacking websites are some of the most common attacks. We look at what goes behind an attack and how one can be stopped.

In the “Network hacking” part we look at hacking network infrastructure and the steps that need to be taken before a successful attack can be made. We divide the process into four steps of “Footprinting”, which is the preliminary research conducted based on freely available information; “Scanning”, which involves poking and prodding network systems for information on vulnerable systems; “Enumeration / Banner Grabbing”, where we actually connect to systems which are attackable and gather relevant system data; “Penetration”, is the final step of exploiting vulnerabilities and constructing attacks based on the information gathered in the previous steps.

In the “Web Application Hacking” part we look at ten of the most common attacks that plague the internet today. The list of attacks is as featured by “OWASP Top 10 for 2010” and we use a framework called WebGoat for studying a few of these attacks.

Over the course of this section we will cover in detail: “Injection”, “Cross-Site Scripting”, “Broken Authentication and Session Management”, “Insecure Direct Object References”, “Cross Site Request Forgery (CSRF)”, “Security Misconfiguration”, “Insecure Cryptographic Storage”, “Failure to Restrict URL Access”, “Insufficient Transport Layer Protection”, and “Unvalidated Redirects and Forwards”.

In concluding, with “Hacking... Anything” we look at how the world of hacking is not limited to only computers. We look at the advantages of hacking and how a hackable application is not always a bad thing.

This Fast Track also includes a few appendices which contain some further information relevant to for those starting their hacking activities. 

1 The basics

1.1 What is hacking?

Hacking is often portrayed to be many things it is not. Thanks to the popular portrayal of hackers as young immoral computer experts associated with nearly any possible illegal and immoral activity that can be conducted through a computer, we see hackers are outlaws of cyberspace, out to steal passwords, or get access to your bank account and steal money. They are portrayed as the equivalent to thieves who break into houses or rob banks – or in the mildest case, peeping toms trying to get a look into your private life.

This could not be farther from the truth. Sure, the act of remotely accessing someone's computer to steal their private files would be hacking. Note the words “steal their private files”, what if that condition was removed? Or what if you are simply accessing your own computer or that of a friend's to help him / her out?

Much like the driver of a car would be called a driver, whether it is done by someone with the car owner's permission, by the car owner himself / herself or without the car owner's permission. Driving is driving regardless of the ethics; the context is irrelevant. Similarly a person is a hacker whether they are bypassing their own computer's security to access their own files, or doing it on someone else's computer without the permission of the owner.

A person withdrawing money from an ATM using their card is okay; a person withdrawing money from an ATM using someone else's card without their permission is a thief. A person hacking into a computer to test its security is a hacker who is a security expert, a person hacking into a computer to steal passwords is a criminal. Here the context decides the legality and ethics of the act, and the person is accordingly labelled a “security expert” or “criminal”, but he is a hacker nonetheless.

Hacking is an expression of our own curiosity, “how does it work?”, “why can't I access it?”, “what happens if I give it 400 volts instead of 220?” It is simply the result of our drive to understand the things around us. Often people are curious about things which may cause them or others harm, such as a child curious about an electricity socket, or an teen curious about drugs. This is no reason to discourage curiosity, the answer, as always, lies in education not restriction. Many of the greatest minds have simply been unsatisfied by the reality they see around them, and looked for ways to “hack” things to work in ways they want. Without their curiosity, and “hacking” skills, where would we have been?

A computer hacker is one who is curious about the working of computers and software. While many people are happy treating their computer like a black box, where they merely feed in data and get data in return, others

strive to break in and understand how it works, and why it works that way. Often, instead of simply accepting things the way they are, they will look for ways to make things work the way they want. While this may be considered juvenile, hacking into someone's computer, just to see if you can, doesn't cause anyone any harm as long as you are responsible enough to respect their privacy.

Don't like the way Windows names shortcuts, hack the registry and change the way it works. Windows may not provide the facility to do this, but that is no reason for us to be limited to the way it works.

In the end, why should there be any kind of artificial limitation to what you can do with your computer? By artificial restriction we mean to say that no amount of hacking is going to make your computer do your laundry! However nothing should stop you from using your computer to its best capacity, as long as it does no one else any harm.

Most hackers are not out to steal money from banks, or crack passwords to sell them, they are there for the thrill of the ride. They will try to hack a system just to see if they can, much like picking their neighbours lock, only to lock it back again – perhaps leave a note telling them they should get a better lock.

Hacking constitutes a mind-set, not a skillset. It's not a "job" it's not something you do for a living. You may earn because of your skills as a hacker, but the hacker mindset is what makes a hacker. Like with anything else, you don't start at the top, you are willing to learn and you poke at things to see where they go; patience is important because it is unlikely you will get what you want in your first try, or your tenth.

To start with, one might simply change the obscure settings accessible to them from Windows, moving further they may install third-party applications which have common hacks for Windows. Then one might go further and change the registry themselves to experiment. The road doesn't need to end here you can start modifying the actual Windows binaries.

Note we say Windows a lot, what about Linux? Fact is, Linux is much easier to hack than Windows. Shock! Horror! Yes, we said Linux is easier to hack than Windows, but it is also considered more secure. When you look at hacking from the larger-sense perspective of messing with a system out of curiosity, Linux allows you to do more. Linux is intentionally hack-able, allowing each and every parameter to be changed by the user. You can create unique combinations of application sets and features that the distribution creators never envisioned. With Linux you have access to nearly all the source code of the system, how much more hack-friendly could it be! With Windows on the other hand, one would need to use third-party tools, patch binaries change undocumented registry settings, and even then the level of customizability would be much less.

In fact, forget Windows and Linux nor now, hacking need not even involve a computer. While hacking is now predominantly associated with computers, hacking hardware is not uncommon. There are many hardware hacking enthusiasts who using some knowledge of electronics and some of software programming are able to bend their devices to their will.

While you can go all the way up to controlling your toaster over the internet, a simple example of a hardware hack anyone can do is to add a potentiometer to your headphones. A potentiometer is an electrical device which lets you control the voltage across a across it. It is a simple way of varying the voltage of a battery (or other power source) from nearly zero to all the way up to the maximum voltage the battery provides. By adding one to your headphones you can control the power of the signal going to the speakers, thereby giving them rudimentary volume control capability. Don't do this with expensive head / ear phones though as you will likely end up deteriorating the quality.

In this book we primarily deal with hacking with reference to computers and information. No single book can employ the broadest possible definition of hacking. After you are done with this book, you will have a better idea of what constitutes hacking on computers. After you are done with this book, you will probably not be able to hack in to others' computers, there are seldom good reasons for doing so, and this book is not for that. This book is meant to fuel a curious mind, and expose it to the world of hackable objects. Before we can begin though we need to address the very subject of this book, ethical hacking.

1.2 What is ethical hacking?

Hacking has been so misrepresented in the mass media that people have had to coin another term "ethical hacking" just to be clear. What does it mean really? Simply that you are a curious person, who likes to mess about with things.

You will never hear the term "ethical baker", "ethical cobbler", or "ethical librarian" but hackers have to go out of their way to ensure others that they are in fact ethical. Every time someone says they want learn how to be an ethical hacker, god kills a kitten¹. If you simply don't have the curiosity then you probably won't want the life of a hacker. And ethics, those you get from you mamma, we aren't the ones to teach you those.

Even so, with the complicated interpretation of ethics, we are left with the question, what exactly is ethical?

Put as simply as possible, being ethical is to not do things which would cause others harm. Popularly the connotation of ethical in ethical hacking is that the person performs his hacking activities within the purview of law. Which arguably might include cases where such hacking in "unethical"

while excluding many instances of “ethical” activities. Let us clarify this with an example:

Imagine someone buys a game. This person has spent money purchasing this game legally, and would like to enjoy the same. However, the gaming experience is continually hampered by the DRM (Digital Rights Management) system which the game uses. The gamer is expected to be continually connected to the internet while playing the game, even though the internet connection is not required for gameplay. His poor connection quality means he will never be able to enjoy the game fully. What if this person then uses his computer skills to subvert the DRM system and play the game directly?

Would this instance of hacking be ethical? It certainly won't be legal. Who is this person harming here? Since this person has already paid for the game, and the game company isn't losing any money the only reason this is illegal is because the law says so, and the law seems to have been crafted with the content creator's interests in mind, not the consumer's. While some may disagree, we shall go ahead and rule this ethical.

The popular meaning of the term “ethical hacker” and the meaning you should derive from it whenever you hear it has to do with computer security – a field where the term ethical is more significant than hacker. In terms of computer security, an ethical hacker is a penetration tester, someone who tries to find vulnerabilities in a system in order to fix them, rather than to profit from exploiting them.

With computer security come two more terms, “White hat hackers” and “Black hat hackers”. These terms derive from old western films, where the villains were usually portrayed as wearing black hats, while the heroes wore white coloured hats.

White hat hackers are those who search for exploits and vulnerabilities in order to fix them, and stop others from being able to hack the system. They do not use their skills in order to harm others or for illegal activities. They are usually hired as security experts.

Black hat hackers are on the other hand, those who hack into systems for malicious reasons, in order to damage and deface web sites, steal passwords, or credit cards. They may do so in order to seek a profit, or out of pure malice. In a perfect world, they would be found in prisons.

There's always a middle ground; that place between black and white where most people live. Grey hat hackers are those who fall in this zone of ambivalent motives. They are those who cannot clearly be placed into the white hat or black hat categories.

As we said before, your ethics are your own, however if you want a career in computer security, your organisation is going to want to be sure that you will not be stealing their money or defacing their web site. There is as much career for a black hat as there is for a professional art thief.

Even as a white hat hacker, no matter how good your intentions, you will not get by simply by memorizing a long list of commands. If your idea of hacking is to memorize all the command and learn when they are applied, then you really are not better than a shell script. You need to be creative, and willing to learn new things.

So after all this we see that the prerequisites for being an ethical hacker are being curious, creative, willing to learn, and of course, being ethical. So why this book?

Well, because we know you are curious, it is why you bought this magazine, we know you are willing to learn as well. We know from your letters, emails and our contests and events that you are creative. You didn't steal this magazine, so you are quite possibly ethical. What are you waiting for? **d**

1 This theory is now disputed. Due to the number of such requests, cats should already be extinct.

2 Information security



2.1 Passwords

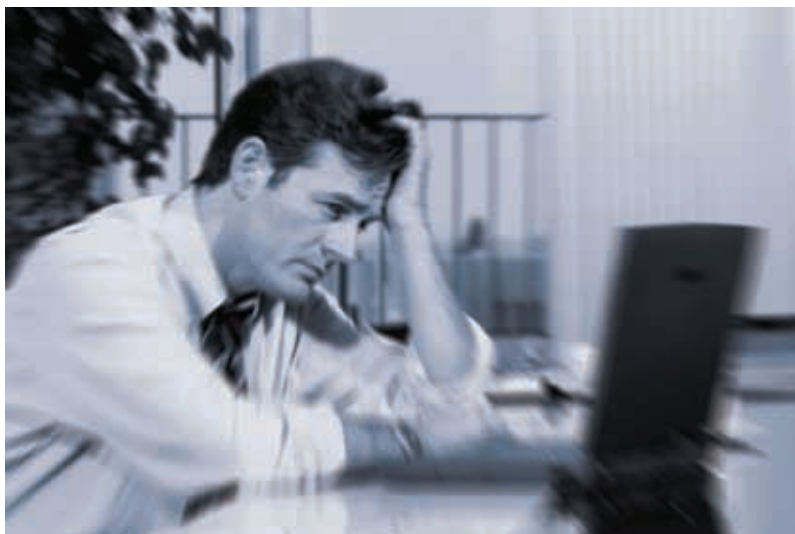
2.1.1 Access Denied

In popular media, “cracking” passwords is often oversimplified. The hacker sits on the computer, mutters a few words about opening sockets and ports, multiple screens light up green over a black backdrop. The hacker somehow manages to “crack” the password by bashing keys on his keyboard – sometimes even multiple keyboards – in what seems to be a random manner while under the influence of fellatio. What impression is one to make of this?

In reality the scene is much different. You can make do with one keyboard and monitor – you won’t be typing all the passwords yourself anyway – and can probably make do without the stimulation.

In reality, hacking passwords is something which requires a lot of research and time. If you are doing some social engineering, it may require considerable work on your part. Finally, it is your computer which does a bulk of the processing work. Chances are you will be twiddling your fingers or catching up to your book reading while your computer is hard at work.

Fact is it isn’t very difficult to make an “impossible” to crack password. Well, theoretically, any password can be cracked given enough time, but when you put in reasonable constraints, you will find that beyond a point, nothing can be done. When we talk to reasonable limits, that would

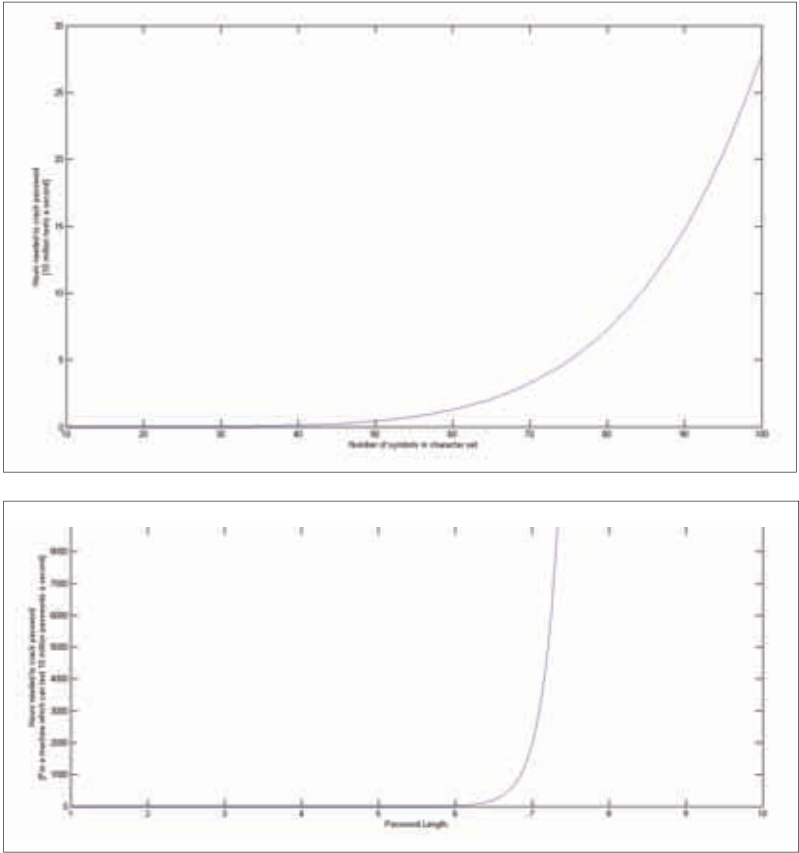


highly depend on the resources one has. Even the latest Intel Core i7 quad core or AMD 12-core processor won't be able to get to a password beyond a certain complexity.

So how much complexity are we talking? What resources? And what are the reasonable limits?

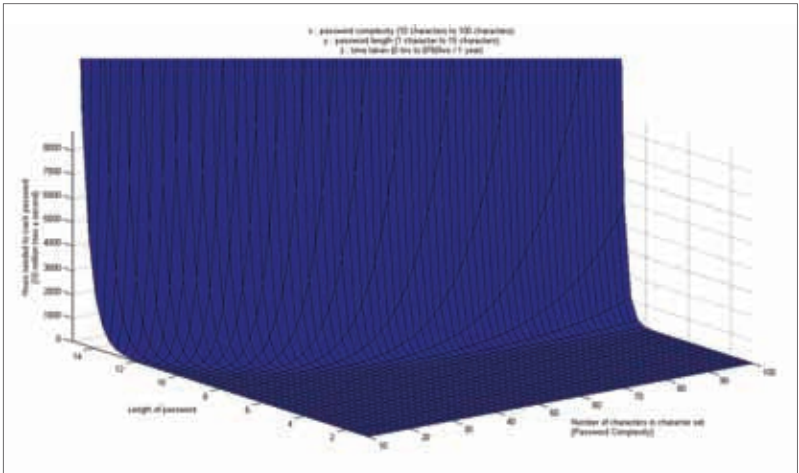
When we talk about complexity, calculating it depends on what all characters are permissible in a password. Let's say we have a password which could be made up of the 26 letters of the alphabet and their capital counterparts, 10 digits, and 32 special characters on a standard keyboard (~ ` ! @ # \$ % ^ & * () - _ = + \ | [{] } ; : ' " , < . > / ?) we have 94 characters, plus the white-space character makes it 95. This means for a single digit password we have 95 possibilities. For a simple two character password, we have as many as $95 \times 95 (= 9,025)$ combinations. For a password of 6 characters, we have $95 \times 95 \times 95 \times 95 \times 95 \times 95 (= 735,091,890,625)$ combinations. For any arbitrary n-character password we can see that there are 95^n combinations. It is important to note that a password cracker will need to test all combinations starting from a lowly single character – if such passwords are permissible – to increasingly long combinations till the password is found. So for password up to n characters, and with a minimum password length of m, the attacker will need to perform $95^m + 95^{m+1} + 95^{m+2} + \dots + 95^n (= \sum_{i=m}^n 95^i)$ tests.

Well, look at it this way; if you have a computer which is capable of testing a million passwords each second, a 6 character password could take as much



as 204 hours! Now the thing here is, that simply throwing hardware on this problem does not help as much as one would like. If we have a computer ten times as powerful, testing 10 million combinations a second, we will still need over 20 hours for a 6 character password, and by simply increasing the number of characters by one to 7, we increase the time required by as much as a hundred times. As you can imagine this situation can easily spiral out of control, as we go for a 12 digit password – which would require about 1.7 billion years for a computer doing 10 million passwords a second – even all of the computers in the world networked together would need a couple of hundred years to crack it! Don't hold your breath.

What we are calculating here is of course the theoretical maximum, and you can expect even with such a crude brute force attack to achieve success



in significantly less time. A real algorithm would take a more probabilistic approach which will check more commonly occurring combinations first. More often than not a password will simply be composed of alphabets and numbers, which decreases our radix from 95 to 62 ($26 + 26 + 10$).

Even so, since the time required guessing a password increases exponentially as the number of symbols in the password increase, we are only buying ourselves enough time for an extra character or two. How about we improve the odds?

In most cases, the passwords people choose for their system will be based on dictionary words. Here of course dictionary doesn't mean an English dictionary, but just a list of words in popular usage which might include words from other languages, common names, slang and 1337speak. This list of words is then used for guessing the password instead of checking each and every combination. As you can imagine, this list will be considerably smaller than all possible combinations, and password length doesn't matter as much as a password's use of dictionary terms. In fact, it might not even matter if the password uses special characters, if such a case is anticipated by the dictionary (for example "cain&abel", "catch-22").

What if, however, the person is using a password which is a combination of dictionary terms with some random characters?

A pure dictionary attack might fail in such a scenario; however, we can use a combination of dictionary and brute force to generate password guesses "near" dictionary words.

The password is not the only point of vulnerability in gaining access to a system / data. When you think about gaining access to something in physical

terms, equating a password protected file to a safe, there are multiple points of vulnerability. It is not enough to make the key too complicated to duplicate if the safe itself is weak; it is not enough to have a complicated password if the encryption algorithm itself is weak. It is important that the safe not have any design flaws which enable someone to subvert the key mechanism. If all else fails, you can simply drill through the safe lock; at this point, there is nothing that can be done to protect the safe as this is equitable to a brute-force attack – in fact this is where the term comes from.

The password algorithm is not the only thing standing in your way either. If you are trying to recover the password for a remote computer or for a system using a authentication mechanism, you need also be aware that the system could limit the number of tries you get to guess the password. For example if the system blocks access to an account after 5 failed password entry attempts, you cannot use a brute force or even dictionary attack.

What if however you simply find the safe key lying written on a piece of paper in a drawer near the safe? Or if it is a physical key, what if it is lying hidden somewhere in the room?

This is where social engineering kicks in, where instead of hacking away directly, we try first to get as much information as possible about the person being hacked in order to make more reasonable guesses. The art of gathering such information and making use of it, is called Social Engineering.

2.1.2 Social Engineering

“There is no patch for human stupidity”

The simplest way to describe Social Engineering would be that it is a means of gathering information about a target using social means rather than purely electronic means. Social engineering works by exploiting bugs and vulnerabilities of the human mind instead of just those of computer systems. A “social engineer” tries to get the target to divulge as much information about them as possible in order to improve their chances at guessing the victim’s password.

Actually, such a technique could be used for more than just guessing passwords. Con artists often rely on learning about their victim in order. A non-password-guessing usage would be to con a person by impersonating as a family member or a friend. Let us look at an example:

A con artist gains as much information as possible about a child in a school, finds out a child’s friends’ names, their parents’ names, the teachers name, and as much more information as possible. This person then visits the parents of said child, pretending to be someone of authority from the school, asking for payment for a new initiative by the school.

This oversimplified example perhaps has many vulnerabilities in itself, however it does illustrate a point. Social engineering is one of the biggest

threats to the security of a system, and as we put more personal data about ourselves out there, it becomes easier to gather this information.

We are often misled by the convenience of the social networking, and the fact that it seems much less personal. Most people give much less thought to making friends on social networks than they do in real life. For some it is about boasting of having more friends, others may feel it rude to reject such an innocent request. It's simpler to add friends online, even those we don't know; especially since few see the harm in this.

The fact is, the second you add someone to your friend list, they might instantly gain access to the kind of information you might not even share with your "offline" friends. Your favourite movies, songs books, interests, hobbies, your list of friends, perhaps even your phone number, address, birth-date, and nicknames are shared with your friends on social networks. All this information is something that your offline friends might only get to know over years of interacting with you, however on an online social network people simply hand it over on a simple request. It does seem to be that online friendships are more serious than offline ones.

You might wonder what the point of social engineering is. How does it relate to hacking passwords? Well the fact is, that most people choose passwords which they find easy to remember – no surprise there – and while not everyone will be choosing passwords such as "abcde" most will rely on phrases of personal relevance, names of members in their family, or friend



circle, important dates, names of pets and so on. Considering this, each bit of information that a hacker can piece together about their target will bring them closer to finding the password.

With this information at hand, a hacker can construct a dictionary of terms related to the target to improve their chances of getting to the password in a reasonable amount of time. Brute-forcing your way to the password is a time-consuming process, and can take days or even weeks. On the other hand, a little time social engineering, and you might possibly find your password in significantly lesser time. Why bash your way to the front door when there could be a key lying under a pot nearby?

So how does one do some social engineering? Here are some common techniques:

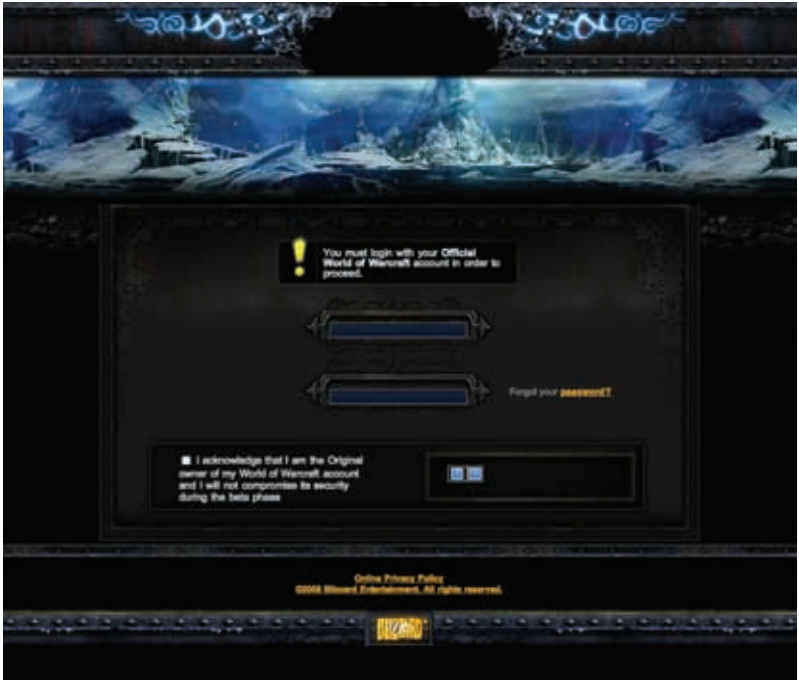
Shoulder Surfing: As you can probably understand from the phrase, shoulder surfing involves finding out the password by actually seeing someone type it in.

Dumpster Diving: It is literally the act of going through someone's garbage in the hope of learning something. Few people take care to properly dispose of sensitive documents.



Impersonation: We gave an example of someone impersonating a school authority to con a student's parent. In a similar manner, one might con an employee of an organization by impersonating a system administrator and point blank asking for the password. Similarly by impersonating customer service, one can over the phone convince people to give up their personal data.

Phishing: Conceptually phishing is quite similar to impersonation, except that instead of impersonating a person, you “impersonate” a web service. A crafty person could, for example, request people to fill in their credit card details for a fake shopping website which mimics the actual version. This is quite simple as it is merely a matter of copying the HTML code. There are many email scams, in which a party claiming to be your email service provider, or bank will try to coerce you to give them your password or credit card information.



2.1.3 The ethical bit

Cracking passwords is interesting sure, but what possible reason could there be to learn social engineering if you want to be an ethical hacker? If you are not going to try getting into someone else's personal data, then why learn how it works?

The answer is simple, as with all other techniques, you need to know how the game is played if you want to win. You may not need social engineering in order to construct your own attacks on other people's files or computers, but you need to ensure that such attacks do not work on you

either. To understand how to protect against these vulnerabilities, you need to recognise them.

The best way to cut off a social engineering attack is to remove the social element. Keep a password that is disconnected from any personal information of yours.

Of course, that alone is not enough, a password such as “password” might have nothing to do with your personal information, yet it is one that is trivially easy to crack, being a dictionary word.

The goals when creating a secure password should be to create a password which:

- Is long (at least 8 characters, 12 or more recommended)
- Does not use a dictionary word
- Is mixed-case
- Contains at least one digit
- Contains at least one non-alphanumeric character

While all this will create a password that is difficult to crack, it will also make it difficult to remember. There is no point to choosing a very secure password if in the end it is too difficult to remember, and requires you to write it down and keep it in your desk drawer. It is hard to say what is the lesson here, whether you should memorise your password better, or choose an easier to remember, but comparatively less secure password.

A good way to have your cake and eat it too is to create a complicated password based on some memory trick. Instead of remembering the password, you could associate it with something easy to remember in itself – no not your personal data! You could go about this both ways, either take a randomly generated password and create a trick to remember it, for example:

`n<7Plc8c` could be memorised as “no less than 7 People like chocol8 cake”

The sentence itself does not make sense, but that might just make it easier to remember! Alternatively, you can take a meaningful phrase and make it into a complicated password.

“When 900 years old you reach, look as good you will not” can become: `w9cYrlgu!` Or “when 9 centuries You reach look as good u not”

This beats any social engineering attack based on your personal information, since that will not be useful in determining the password. While an attacker might be helped by learning of your liking for “Star Wars”

with the number of awesome Star Wars quotes, and the passible passwords that can be made from them, just know that you are safe!

We still have not addressed how to be wary of shoulder surfing and phishing. To counter those, the best way is to be more careful of your surroundings and ensure the authenticity and security of the websites where you enter your personal data. For shoulder surfing, there is one trick that can make your password difficult for a casual observer to guess: Use a double character in your password. If a password has a character repeated twice, it will be difficult for an observer to notice the repetition since your fingers won't move from the key. If you type your password fast enough, most people won't be able to detect a character pressed twice.

Another common flaw in most people's password policy is to use the same password for all their accounts. This is quite understandable, most people have a large number of accounts, and it would be difficult to remember a unique password for all. You can however put a little extra effort to make data at least a little bit more secure. For one, you can choose different passwords for your more important accounts such as online banking and lesser important ones such as the website accounts you may have created just to leave comments. This way, even if you password on one of the less secure websites is cracked, your important accounts will be secure.

Another thing you can do is to choose a base password and modify it depending on the website you are using. If your base password is "x#f33todr" you can create a Facebook password of "Fx#f33todrb" or something similar. If you maintain the same pattern everywhere your password will be easier to remember. However it will also be easier to guess for someone who knows your trick – but you won't tell anyone will you?

Passwords are the fragile little things holding our online life and the security of our organizations together. They are the short character sequences that lie between us and total destruction. Keeping a secure password is very important, and it is equally important to understand how you might be at risk. I hope that this section has been of help.

Next, we talk about Hashes, the short character sequences that strive to keep our password secure despite adversity.

2.2 Hashes

2.2.1 What the #!

When we open an account with any service provider, we are first of all giving them the password to our data. What happens then if a disgruntled employee leaks this data, or if the account data is exposed in case of software vulnerability?

The problem with passwords is that somewhere or the other you need to store the damn thing just so you can access it later on to authenticate a login



attempt. If the file is stored somewhere on the disk there will be a way to access it and the game is over before it begins!

You might be thinking that such a file could simply be encrypted, but then you'd just have another password to save somewhere!

This is where hashes come in. Let us first understand what a hash is.

A hash is comparable to a person's fingerprint. While the actual authority lies with the person himself/ herself, for all practical purposes the fingerprint of the persons is considered enough to establish a unique identity. While it is theoretically not impossible that two people have the same fingerprint, the chances of such a thing happening are rather slim.

Similarly, the hash of any data is a fixed size "fingerprint" of that data. If we have the hash of a piece of data – say a password –, it is not possible to get back the original data. How then will such a hash help in securing passwords, or even exposing them?

A hash function is such that given the same data; the computed hash for that data will always be the same. Therefore, if an application chooses to save a hash of a user's password instead of the password in plain text, the application can check if the password a user enters is correct by checking if its hash matches the stored password hash. A weak analogy to a real world case would be, you cannot get back milk from curd, but you can check to see if a white fluid is milk by seeing if it curdles.

A hash is different from encryption, since encryption by its very nature has to be reversible, which a hash is not. A hash is not reversible, and as such, even if one finds out the hash of a user's password, they will be unable to reverse the hashing process to recover the original password. This is not

to say that having knowledge of a user's password hash is useless, it isn't. It just so happens to be that computing the original password given the hash is not a trivial task.

An authentication system which uses password hashes would work as follows:

Registration:

- User creates an account using a particular password.
- The registration system stores the password hash, not the password in the database.

Authentication:

- User tries to log into the website with a password
- The application computes a hash of the password the user enters
- The application checks to see if this hash matches the saved hash for the password.
- If they match, the user is authenticated.

Let us take a very simple example of how this might work.

Let us take a security mechanism of an application that allows only 6-digit numeric passwords – such as those found in ATMs. This security system takes a user's password, and stores it as a hash of the original numeric password. That hashing function derives the hash of a number by taking the average of pairs of numbers in the original password.

So the password: 864159 [password]

Would become: 737 [hash]

Derived as:

$$\frac{8+6}{2} = 7 \quad \frac{4+1}{2} = 2.5 \simeq 3 \quad \frac{5+9}{2} = 7$$

Now, while it is very easy and fast to calculate the “hash” here, it is not possible to get back the password from this hash. However, as long as the user is inputting the same number, the resulting hash will be the same. Therefore testing hash of the number is as good as testing the original.

Since the “hash” is much smaller than the password, it is clear that many passwords will share the same hash. This kind of scenario is quite common with real hashing algorithms, which produce hashes of a fixed length. It is called a collision, and while developing a hashing algorithm, great care is taken to ensure that one cannot derive two inputs that produce the same hash easily. Even the smallest of changes in the input should alter the hash. Real-life hashing algorithms are significantly more complicated than the example described here of course!

A popular hashing algorithm is MD5 (Message-Digest algorithm 5) which always produces a hash of 128 bits. So for any input, whether is a 3-character string, or if it is a video file of a few gigabytes, the hash which MD5 will produce will be just 128-bit long. Another popular hashing function SHA-1 (Secure Hash Algorithm) produces 160-bit hashes. SHA1 has been superseded by SHA-2, which has four function that produce hashes of 224bit, 256bit, 384bit, or 512bit.

MD5 for example will create hashes such as the following:

```
"password"      : "5f4dcc3b5aa765d61d8327deb882cf99"  
"iddqd"         : "73bcaaa458bff0d27989ed331b68b64d"  
"iddqD"         : "93cbd5c967c1d2882aadae3950ea22be"
```

The following are examples of SHA-1 hashes:

```
"password"      : "5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8"  
"iddqd"         : "67de5036a631cad20dbfe5600d6b0060fa0ce03a"  
"iddqD"         : "07735670770727d25c67d1879d5733724ec4e4ca"
```

As you can see, even a small change in the input string (from "iddqd" to "iddqD") results in a completely different hash.

Hashes can be used for other purposes than just securing passwords. You might be familiar with their usage by download sites for verifying the integrity of a download. A download site may provide you with an MD5 / SHA-1 hash for the file so that once you have downloaded the file you can verify that the download is complete and intact. Any difference in the hash of the file means that the download is possibly corrupted or malicious. If the file can be downloaded from multiple mirrors, this also ensures that external websites do not provide malicious content with the same name.

Torrent files use SHA-1 hashes of each "piece" of the content you are downloading. In torrents, the download content is divided into sections / pieces of fixed size from 16kB to 4096kB, and each piece of the download is hashed separately. This way if you are downloading a 10GB file – or ten 1GB files – in case of an error / shutdown / crash, only the corrupted piece of the file needs to be re-downloaded.

Hashes have many security related usages, hence their importance to hackers black-hat and white-hat alike. Hashes find uses in message authentication, and digital signatures as well. Hashes are equivalent to fingerprints of digital data, and understanding them can help a lot in hacking and preventing hacking.

In the next chapter, we explore the weaknesses in hashes, how they can be exploited and possible ways to secure them.

2.2.2 Of Rainbows and Salt

Once you have the password hash, then what? How can you get a user's password from a given hash? How can you exploit hashes to compromise security? Most importantly, how can you secure against such attacks.

To reiterate the way password hashes work, when you login, it is your password that is used to authenticate you, however instead of using your password to authenticate, it uses the hash of the password.

Example:

User's selected password: qb7%R1e

Stored password hash (MD5) : 4f55ffad7d8c44c2538ccaf5b5ef407

User provides the password: h6Sk3ty

Hash for password: a444724e17a3a3a5704a6fb81db4dd60

Authentication fails

User provides the password : qb7%R1e

Hash for password : 4f55ffad7d8c44c2538ccaf5b5ef407

Authentication succeeds

A very simple way of recovering a password from a hash would be to apply the same techniques we do to password cracking. This is, to generate possible password candidates, hash them, and see if the hash matches. As you can imagine, this will be a time consuming process, although since this process will be offline we will not have to worry about limited retry attempts.

One advantage we have with hashes though is that one can easily create a database of plaintexts and their corresponding hashes. This way, in the future one can simply look up the hash in their database and they will have the password instantaneously instead of needing to wait hours!

By creating an exhaustive database of string-hash pairs one would need to calculate the hashes only once, and in the future this database could be utilized to look up a string from the corresponding hash.

Unfortunately, creating such a database of hashes will require an incredible amount of storage space, and will only work up to a point. A 10 character password as discussed before has 9510 combinations (additionally, smaller passwords will also need to be tested) which is a rather large number (59873693923837890625). Even storing a list of all possible 10-character passwords will take 544548074 TB of space! With the hashes, the storage requirement will become 1415824992 TB. While these kinds of storage capacities are not technically impossible, they are currently not feasible.

Even so, it might be feasible to have a database containing hashes of all dictionary words, and variations thereof. However, dictionary attacks are already much faster than brute-force attacks, and this just means that the

power of having a database of hashes is not fully being exploited. An attack using such a database would consume too much memory, and a direct attack would be too time-consuming. The trade-off here is between too much time (days, months, years), or too much space (TB, PB, XB). If only there was a middle ground, which used lesser space than a fully exhaustive database of hashes for all possible strings, and one which would take significantly lesser time than a brute-force attack.

This is where rainbow tables come in. Before going in to rainbow tables, let us first understand their predecessors, hash chains.

The concept of hash chains allows for a good compromise between the speed of using a hash database and the memory usage of bruteforcing. Hash chains allow for saving significant amounts of space by finding out the hashes for limited number of texts and deriving the rest from them.

Hash chains rely on two functions, the hashing function and the reduction function. While the hash function – as expected – maps a plaintext to a hash, the reduction function maps a hash to a plaintext.

Here when we say that the reduction function maps a hash to a plaintext, it does not mean that it is finding the original plaintext that created the hash, but a plaintext that is derived from the hash. For those who remember sets in mathematics, a hash function is a function that maps the set of all plaintexts to a set of all hashes; a reduction function is a function that maps the set of all hashes to a set of all plaintexts. The plaintext that the reduce function will return will not be the same one which was used to create the hash; however it will be a valid plaintext nonetheless.

Example:

The mathematical function:

$$h(x) = 2x + 1$$

Operates on a set of all integers, and maps the set of integers to a set of odd numbers.

Another function:

$$r(x) = \frac{x+3}{2}$$

Operates on a set of all odd numbers, and maps the set of odd number to a set of integers.

Here, the function $h(x)$ will always map to an odd number (comparable to the “hash”), and given this hash, the function $r(x)$ function will always result in an integer (comparable to the “plaintext”). However, it is important to note that the reduction of a hash will not result in the original plaintext.

In mathematical notation:

$$r(h(x)) = x + 2 \neq x$$

Moving on.

Now that we have a hashing and reduction functions, what we do is create a chain of hashes and plaintexts. We pick a plaintext to start with, hash it, then use the reduction function to get another plaintext from the hash, this plaintext is again hashed, and then reduced, and then hashed, and so on for a specified number of steps.

What we have here is a chain of hashes and plaintexts that can all be derived from the starting plaintext. Such a chain could go on for thousands of iterations.

Now instead of storing each and every plaintext-hash pair along the chain, we only store the first plaintext and final hash, since all values in the middle can be computed by repeatedly using the hash and reduction function on the original plaintext.

Such a chain of say 1000 is storing information about 1000 hashes while using up only the amount of memory required for storing one plaintext-hash combination! We can now create thousands or millions of such chains and be able to cover nearly all plaintexts, while using up only a thousandth of the space.

It will still not be feasible of course to have plaintext lengths of 10-20 characters, however this method is good up to 8-9 characters – maybe even more, depending on the algorithm, the length of the chains, compression and other factors –, which depending on the hashing function, chain lengths, character set etc. can leave you with a file of a few GBs.

Let us create a chain of length 5 with a hashing function of MD5 and a reduction function which simply takes the first 6 characters of the hash as the plaintext. We will start with a randomly generated 6-digit alphanumeric plaintext:

Plaintext	:	a6tyv2
Hash	:	cd5b5e698eafeda22ab0370a88f79410
Reduced result	:	cd5b5e
Plaintext	:	cd5b5e
Hash	:	f945501a94a6836b5be04ba8fba908b0
Reduced result	:	f94550
Plaintext	:	f94550
Hash	:	e4ff786adec3a903dc3073720230117d
Reduced result	:	e4ff78

Plaintext : e4ff78
Hash : c1b2d3114d814f41638bdafcaeb8e057
Reduced result : c1b2d3

Plaintext : c1b2d3
Hash : 6e929c2625cc49bc50d2d1ae845f9cf7

This chain will be stored as a combination of “a6tyv2” and “6e929c2625cc49bc50d2d1ae845f9cf7”

So how will this help with cracking hashes?

Here is how it will work. When you are looking for a plaintext for a particular hash, the cracking algorithm will first look to see if the hash exists in the table. If it does, we know that the chain for the matching hash contains the plaintext. If we don't find the hash, we reduce the hash to a plaintext and hash that again. We then look for this hash in the table. We continue this process of reducing and hashing until we find a matching hash. Once we find a matching hash, we know that the chain for the matching hash will contain the plaintext.

Let us see this in play using the data in the example given before. We are looking to crack “e4ff786adec3a903dc3073720230117d” and get its plaintext. Our database contains just one lonely entry:

a6tyv2 : 6e929c2625cc49bc50d2d1ae845f9cf7

Check if the input hash matches a hash in the database.

Result: Hash not found.

Reduce the input hash (e4ff78), and hash again.

Result: c1b2d3114d814f41638bdafcaeb8e057

Check if the hash matches a hash in the database.

Result: Hash not found.

Reduce the input hash (c1b2d3), and hash again.

Result: 6e929c2625cc49bc50d2d1ae845f9cf7

Check if the hash matches a hash in the database.

Result: Hash found.

Now that we know that the hash belongs to this chain, we start with the initial plaintext (a6tyv2), reduce, and hash our way through the chain until we find the plaintext that gives us our hash.

Such hash chains have a few problems though. They suffer from the defects of merging chains and cyclical chains. If we happen to start with a plaintext which at some point in the chain results in a plaintext / hash included in another chain, then after that point, the chains will “merge” and produce the same result. This causes inefficiencies in storage. A cyclical chain will result when a hash along the chain results in the plaintext we started the chain with.

Rainbow tables counter such defects.

In a rainbow table, the reduction function used is varied across the iterations. This makes looking up hashes more complicated, as they need to be tested with each reduction function, but the chances of branches merging are negligible and cyclical chains are not formed.

Now that we know how rainbow tables work. Let us look at the way to counter them Salts.

A salt is just a randomly generated non-secret sequence that is added to the password to make it truly unique. This way even for two users with the same password, the hashes will be different. If we add a simple single character salt to the original plaintext, we get a drastically different hash.

If a hacker is prepared with rainbow tables up to 7 alphanumeric characters, the addition of a salt of a single character to the 7 will make the rainbow tables useless. Of course, hackers can be better prepared in the future and have rainbow tables pre-computed for all possible salt values provided the salt is small enough.

Older UNIX passwords were stored with a 12-bit hash that meant that the passwords could be cracked if the hacker was prepared with a table for each of the 4096 possible salt values. With hard drive prices getting cheaper, hackers were able to create tables for all salt values and store them, making it possible to crack such passwords.

A rainbow table for a MD5 hash for lowercase-alpha-numeric passwords is around 3GB in size. For a 12-bit salt, this would jump the size to 12TB! This could perhaps be compressed to around 4TB; such storage capacities are available nowadays.

However a simple increment of 2-bits for the salt, and we will need 4 times as much space. So if a salt of 48 bits to 128 bits is used – as it is in current *nix OSs – the hash becomes impractical to crack.

2.2.3 Cracking a hash with rainbowcrack

The first step is to generate the rainbow tables, which is done using the utility “rtgen”. We need to provide some configuration options such as the hashing algorithm the character set, the password lengths, etc.

The parameters are as follows:

rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index chain_len chain_num part_index

hash_algorithm

The hash algorithm to use for generating the tables md5, ntlm / lm for Windows

charset

The character set for the password. Supported options are:

numeric	0123456789
alpha	ABCDEFGHIJKLMNOPQRSTUVWXYZ
alpha-numeric	ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
loweralpha	abcdefghijklmnopqrstuvwxyz
loweralpha-numeric	abcdefghijklmnopqrstuvwxyz0123456789
mixalpha	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
mixalpha-numeric	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
ascii-32-95	!"#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{ }~
ascii-32-65-123-4	!"#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`{ }~
alpha-numeric-symbol32-space	ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#\$%^&*()-_+=~[]{} ;\:;'"<>.,?/

plaintext_len_min plaintext_len_max

These are some of the more obvious parameters. They define the minimum and maximum lengths of the plaintexts for which the password is to be found. So for a length from 1 to 7 there will be no likelihood of there being a hash for a 8 character password, but most combinations within the specified character set will likely be present.

table_index chain_len chain_num part_index

These are the least obvious and most complicated parameters. Since you have some idea of rainbow tables, you will probably know what chain_length means. Hint, it is the number of hash-reduce cycles which form the chain, and lead to the final hash stored for each chain.

The table_index is used to define which reduce function to use for the table and the part_index is for deciding how to generate the initial starting point for the rainbow table.

The settings recommended by the creators of the software for a 99.9% probability of cracking hashes are (for md5):

rtgen md5 loweralpha-numeric 1 7 0 3800 33554432 0

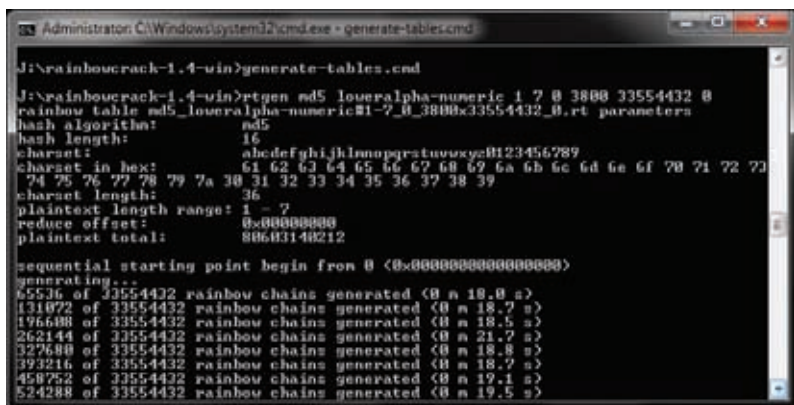
Instead of md5, you could have used any other algorithm here. Rainbowcrack can be extended using dlls to support any hash algorithm.

Here we are generating a rainbow table with the following specifications:

hash_algorithm	md5
charset	loweralpha-numeric = [abcdefghijklmnopqrstuvwxyz0123456789]
plaintext_len_min	1
plaintext_len_max	7
chain_len	3800
chain_num	33554432

This command has to be run 6 times for each table index, from 0 to 5, and each time it will take around 2hrs on a Core i5 system! Each resulting file will be around 512MB, making the whole table collection around 3GB in size.

As you increase the number of characters in the character set and the length, the size will go up drastically.



```

Administrator C:\Windows\system32\cmd.exe - generate-tables.cmd

J:\rainbowcrack-1.4-win>generate-tables.cmd

J:\rainbowcrack-1.4-win>rtgen md5 loweralpha-numeric 1 7 0 3800 33554432 0
rainbow table md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt parameters
hash algorithm:      md5
hash length:         16
charset:              abcdefghijklmnopqrstuvwxyz0123456789
charset in hex:       61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
                    74 75 76 77 78 79 7a 30 31 32 33 34 35 36 37 38 39
charset length:       36
plaintext length range: 1 - 7
reduce offset:        0x00000000
plaintext total:       00603140212

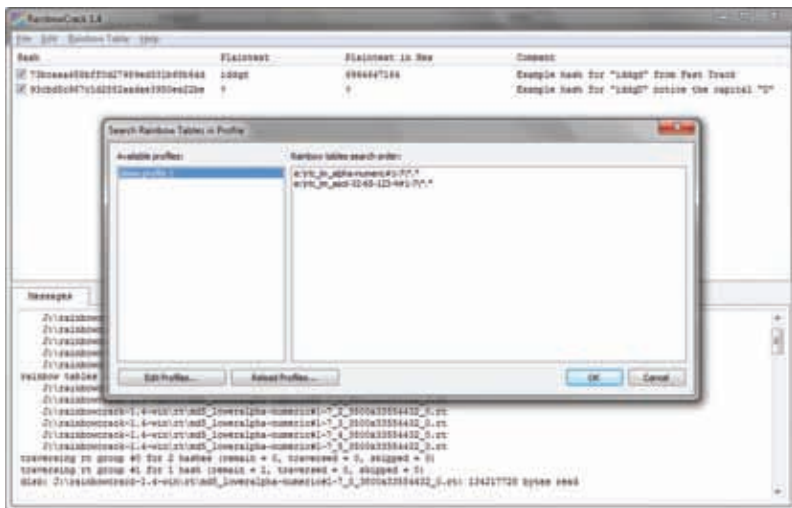
sequential starting point begin from 0 <0x0000000000000000>
generation...
65536 of 33554432 rainbow chains generated <0 m 10.0 s>
131072 of 33554432 rainbow chains generated <0 m 18.7 s>
196608 of 33554432 rainbow chains generated <0 m 18.5 s>
262144 of 33554432 rainbow chains generated <0 m 21.7 s>
327680 of 33554432 rainbow chains generated <0 m 18.8 s>
393216 of 33554432 rainbow chains generated <0 m 18.7 s>
458752 of 33554432 rainbow chains generated <0 m 19.1 s>
524288 of 33554432 rainbow chains generated <0 m 19.5 s>
  
```

Running the rtgen tool to generate rainbow tables

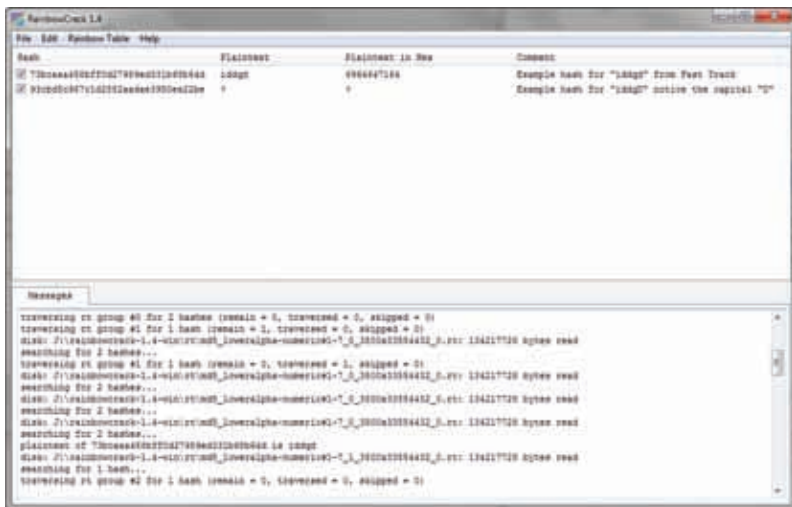
This will result in files named “md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt” to “md5_loweralpha-numeric#1-7_0_3800x33554432_5.rt”

The resulting rainbow tables will not be sorted, and hence will not be easy to search in. So the next step is to sort the tables using the rtsort command. This command simply takes the filename of the unsorted table as a parameter. This takes under a minute to execute.

Since rainbow tables are large, you can also use rt2rtc command to compress the tables into “.rtc” files which can be used with the cracking application without needing to be decompressed.



If you use the application often enough, you can create profiles for your rainbow tables to quickly load them.



In less than a minute after selecting the rainbow tables, we have the decrypted plaintext for the hash. We used the hash for "idqd" as given before. A few minutes later the application fails to find a plaintext for the "iddqD" hash, since we only generated tables for lowercase.

```
Administrator C:\Windows\system32\cmd.exe
J:\rainbowcrack-cuda-090817>rcrack_cuda.exe J:\rainbowcrack-1.4-win\rt\*.rt -h cd5b5e698eafeda22ab0378a88f79410
traversing rt group #0 for 1 hash (remain = 0, traversed = 0, skipped = 0)
traversing rt group #1 for 1 hash (remain = 0, traversed = 0, skipped = 0)
disk: J:\rainbowcrack-1.4-win\rt\nd5_loweralpha-numeric#1-7_0_3800x33554432_0.rt : 268435456 bytes read
searching for 1 hash...
disk: J:\rainbowcrack-1.4-win\rt\nd5_loweralpha-numeric#1-7_0_3800x33554432_0.rt : 268435456 bytes read
searching for 1 hash...
plaintext of cd5b5e698eafeda22ab0378a88f79410 is a6tyv2
disk: thread aborted
GPU #0: max cuda kernel runtime is 0.19 s

statistics
-----
plaintext found:                1 of 1
total time:                     8.81 s
  time of chain traverse:       1.36 s
  time of alarm check:         0.39 s
  time of wait:                 2.45 s
  time of other operation:      4.62 s
time of disk read:              6.57 s
hash & reduce calculation of chain traverse: 14428682
hash & reduce calculation of alarm check: 2694258
number of alarm:                2591
speed of chain traverse:        10.62 million/s
speed of alarm check:           6.91 million/s

result
-----
cd5b5e698eafeda22ab0378a88f79410  a6tyv2  hex:613674797632
J:\rainbowcrack-cuda-090817>
```

The command line is also quite simple to use. Here we find the plaintext for another hash example we have used in this book while explaining chains. We used the cuda enabled rcrack application which took a mere 8.81 seconds to crack the hash and give us "a6tyv2"!

3 Hacking the web / network

3.1 Network hacking

3.1.1 Introduction



The archive.org search for thinkdigit.com shows snapshots of the website from 2001...

Usually, the system you want to hack is not one you have physical access to, and information you need is not on your own computer, but on some remote computer in your local network, or on the internet. Hacking into a remote



... and a year later in 2002

system requires patience and research, since – unlike a computer you have access to – you will probably not know anything about the computer and the software running on it.

Consequently, the most of the steps in hacking a remote system are those that have to do with building a repository of any and all information about the system one wishes to hack. Only once you have sharpened the knife enough, do you think about using it.

You may recall social engineering, which was covered earlier, where in order to guess a person's password, one researches as much information about them as they can. However, passwords are only one piece of the larger puzzle, and only one of the ways of bypassing security.

Even if you have the securest of passwords and the best password policies, malicious hackers will not be deterred. They will find a way into your system, often bypassing all your security by exploiting flaws in your software.

Once a hacker has enough information about the target system, he / she can begin to research the vulnerabilities in the system and use them to construct an attack. As an ethical hacker, it is important to keep note of the vulnerabilities exploited in each step of the way, and patch them.

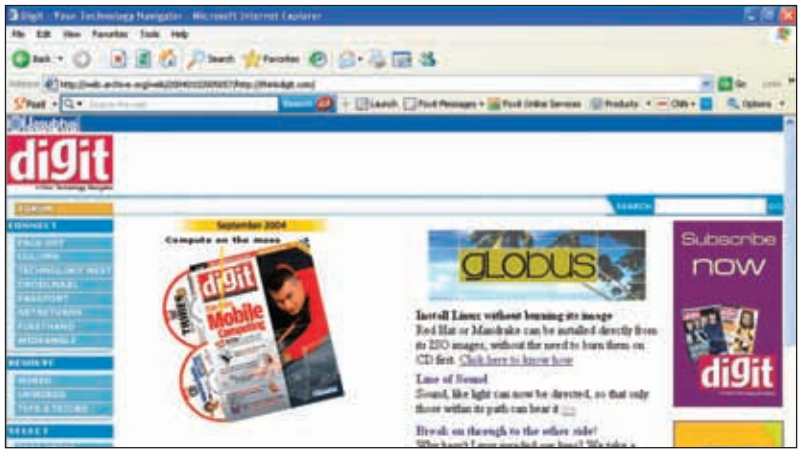
3.1.2 Footprinting

The first step in gathering information about the target is the task of footprinting. Footprinting can be non-intrusive. Here by non-intrusive we mean that the process of footprinting does not rely on actively engaging the target. It is like a detective finding out addresses, and telephone numbers, not knocking on doors, and picking locks. Footprinting is the creation of a profile for the target containing all information about the target's level of security.

The target might be on the local network or on the internet, however in both cases, we try to determine as much information about the system(s) as we can. We will need to find out the IP address or the range of IP addresses which belong to the target, the domain names, the subnet, the services running on the target systems, the operating system and architecture of the systems, and as much more information as is possible.

As we have said before, you cannot blindly attack the system without first knowing what your vector of attack should be. Your goal should be to obtain as much information as possible that any malicious hacker would be able to obtain and use for advantage. Once we have this information, we know how exposed the systems are to harm, and can begin actually hacking the system to test its security.

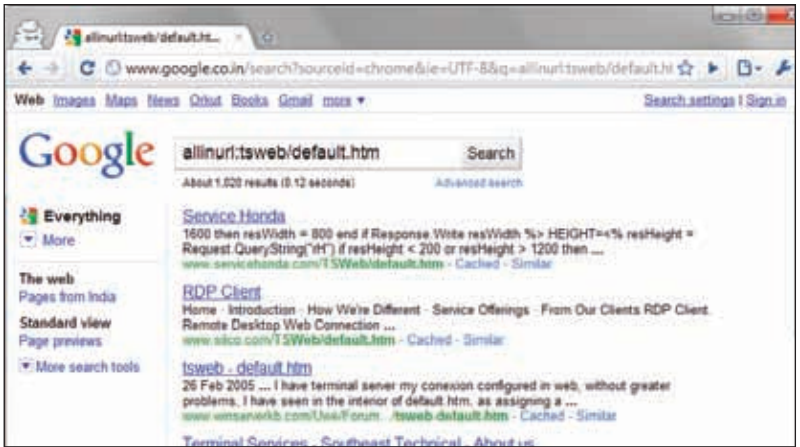
For finding this information, there are a wide number of tools available, but a large amount of this information is simply publicly available. A



thinkdigit.com as it was way back in 2004

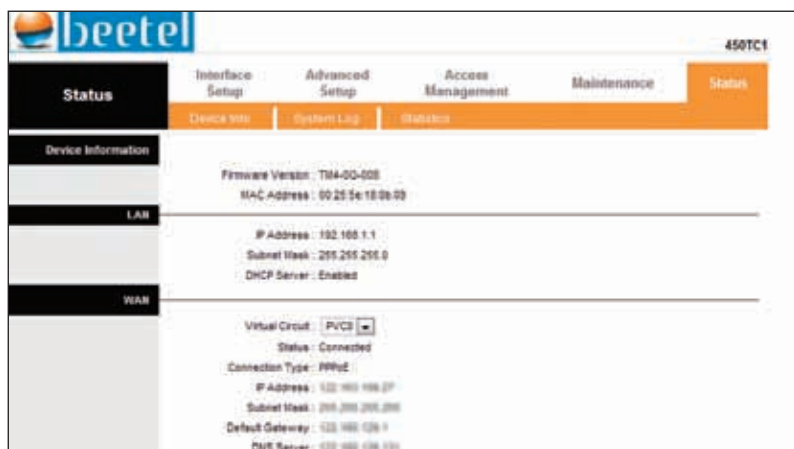
WHOIS record on any domain will get you address, phone numbers, email addresses, and much more. With a WHOIS lookup for a domain – which is free and non-intrusive by the way – one can also get information about the name servers for a domain. Archiving services such as www.archive.org keep snapshots of publicly accessible web sites so that even in the future you can get to see where the website is coming from.

With a large number of people having profiles online, it is not difficult to gain information from such social networks. By having access to the

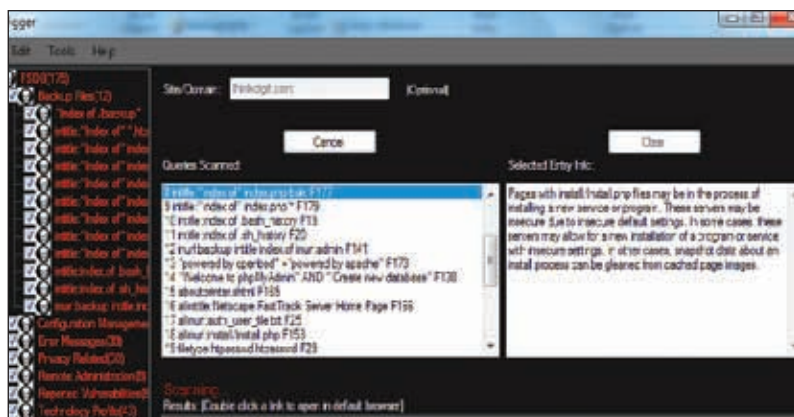


Google reveals the webservers running Microsoft's Remote Desktop Web Connection.

Google is god here, and with a few specific searches can reveal a lot of information which people would rather have kept private. For example, many services or applications running on the target system might provide remote administration interfaces that might not be secured.



A router configuration page can give you access to the username and password



Site digger at work scanning a website for vulnerabilities

For example in India, a large number of people access internet through ADSL routers. These routers offer a web configuration interface usually hosted on “192.168.1.1”, however often this interface is also accessible from the router’s external IP address. So by scanning the list of MTNL IP addresses for example, one will run into many people who have their routers set up insecurely and can be configured remotely through the browser. Some of these might just have the default username and password. By accessing such an IP through the browser, one will be able to reconfigure the router and get a peek at the username and password of the end user.

```
C:\>tracert www.twitter.com

Tracing route to twitter.com [128.242.240.20]
over a maximum of 30 hops:

  0  1 ms  1 ms  1 ms  192.168.1.1
  1  96 ms  202 ms  *  ABIS-North-Static-001.129.160.123.airtelbroadl
  2  *  *  144 ms  ABIS-North-Static-249.242.160.123.airtelbroadl
  3  *  *  *  Request timed out.
  4  203 ms  *  203 ms  AES-Static-114.36.144.59.airtel.in [59.144.36.
  5  *  *  *  Request timed out.
  6  *  475 ms  489 ms  ge-2-1-0.r03.engps102.sg.bb.gin.ntt.net [129.2
  7  *  *  *  Request timed out.
  8  463 ms  473 ms  436 ms  ae-2-r20.enjsca04.us.bb.gin.ntt.net [129.250.2
  9  *  *  487 ms  ae-2-r20.nlpcca01.us.bb.gin.ntt.net [129.250.5
  10  *  *  *  Request timed out.
  11  580 ms  526 ms  547 ms  128.241.122.197
  12  450 ms  549 ms  535 ms  128.242.240.5
  13  513 ms  607 ms  543 ms  128.242.240.20

Trace complete.

C:\>
```

The road to Twitter is paved with 10 hops

There are applications such as SiteDigger, which include a large database of such common vulnerabilities in web sites. The SiteDigger application allows you to select from a list of vulnerabilities to test a web site for, and gives detailed information about the same.

Performing a trace-route is another way of getting information about the points along the way to a web site. The “tracert” command reveals information about the hops on the way to accessing a domain, and with this path revealed, we could have a better idea of how the system connected to a network. For example what the last hop is to the system itself, you can expect the second last hop to be to a router or firewall to control access to the server.

All this information is invaluable when constructing an attack, and it is important for us to know so we can know how to prevent such information from being abused. Often you will be surprised by the volume of intimate information available about your own organisation online.

3.1.3 Scanning

We can consider footprinting the equivalent of taking images of a bank safe and finding out its make and model, and noting the routes of the bank guards, finding out who has the key. Scanning is more intrusive, and involves actually interacting with the target systems. In this step we would follow the “footsteps” and do some actual testing.

Now that we have an idea of the scope of the network that we need to infiltrate, we start poking and prodding at the computers on the network looking for systems that are vulnerable. Specifically, we look for systems that are “alive”, i.e. listening for input on some or the other port.

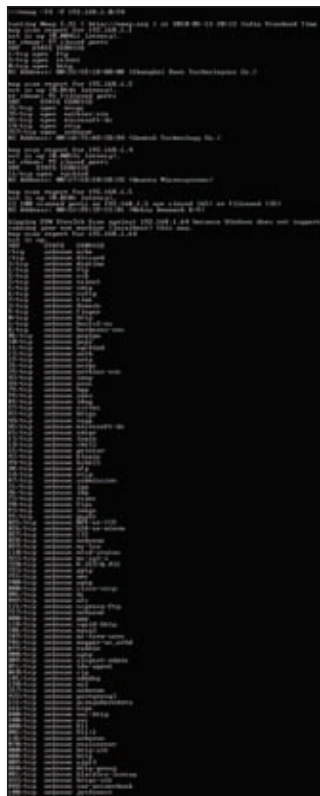
By using a mass ping utility, one can get an idea of which systems on the network are alive, so that we can further test for security lapses.

A more comprehensive look at the systems on a network can be obtained by using utilities such as “Nmap”. This handy tool is a command-line utility that can do everything from a simple ping sweep to a fully comprehensive scan of all open ports on a system.

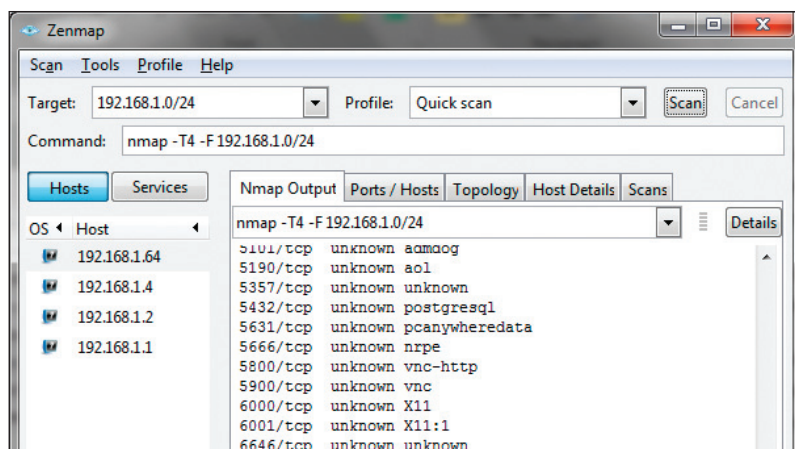
ZeNmap is an easy to use GUI frontend for Nmap that lets you browse the information provided by Nmap in a much richer way. It includes common network testing profiles that define how much is scanned. You can provide it an IP range to scan and it can construct a topological map of all connected systems in the network.

A quick Nmap scan will reveal all open ports in the system, and as far as possible, it will also reveal the services running on the port. With ident scanning, Nmap will also reveal the owner of each running service – as in the privileges the service is running with. Knowing the identity of the user can help with the attack, as by attacking a service which is running with a privileged (i.e. root) account we stand to gain full root access.

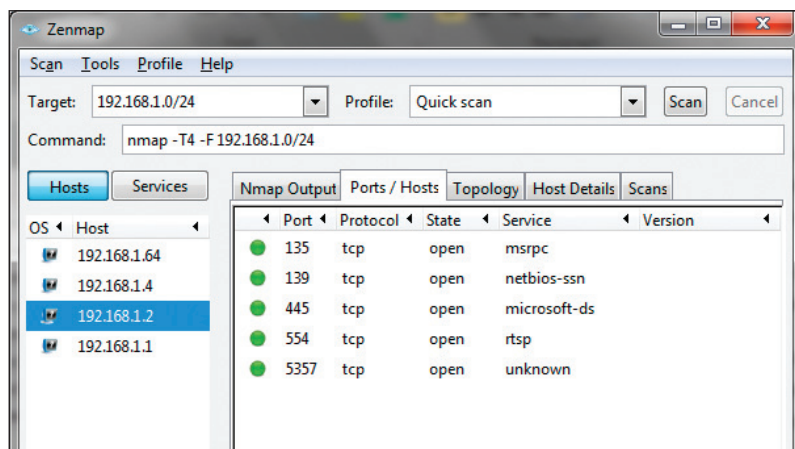
We have used Nmap exclusively in our examples till now, however this does not mean it is the only tool, or even the only good tool. Strobe, Netcat, SuperScan, and many more applications are available which can perform



nmap running in the Windows console, scanning all IPs in the local network



Zenmap showing the output from nmap for a scan of computers in the local network.

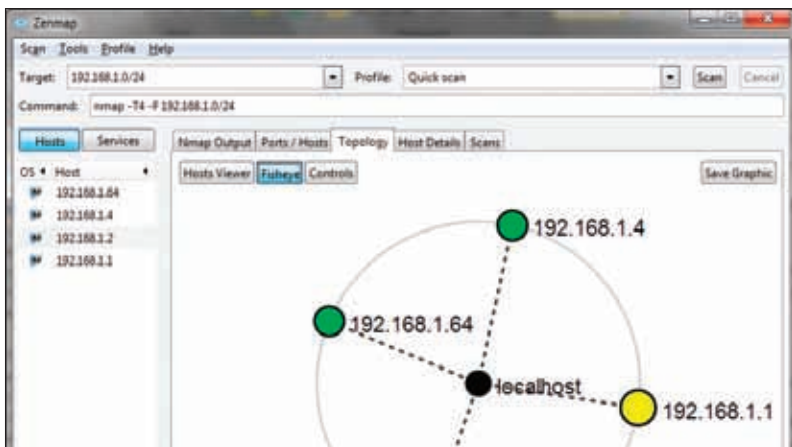


Zenmap showing a list of ports on the selected host (192.168.1.2)

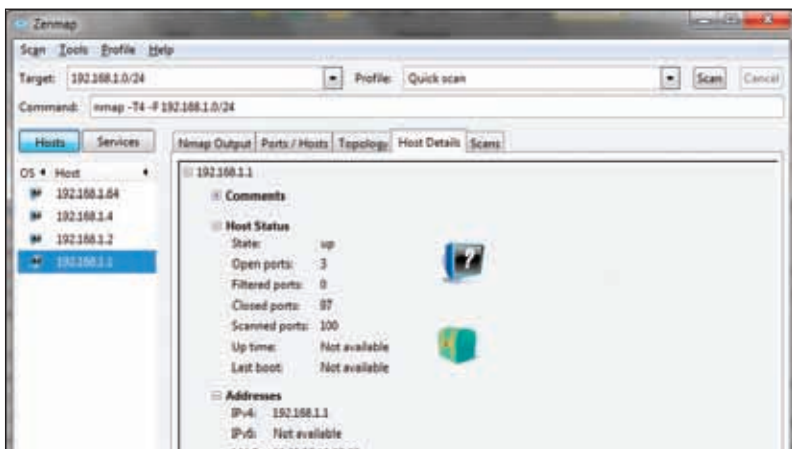
some of the above-mentioned tasks. While there are many other tools for performing such scans, Nmap is one that works across platforms and supports the maximum number of features.

With the use of these tools, we have an idea about what all services are running on which system and on what ports, we know which OS each computer is running, and possibly the privileges of the running services.

Armed with this information, one can understand further areas where the security of the system can be improved. It is wise to disable any



Zenmap showing the topology of interconnected computers on the network.



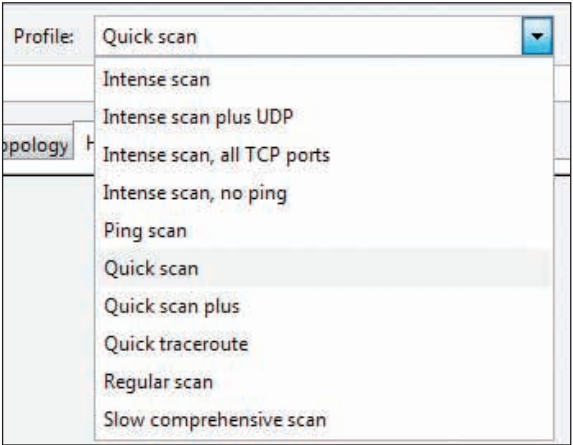
Zenmap showing details of the selected host

unnneeded services, and to ensure that the others are running without root or administrative privileges to dampen the extent of any hack.

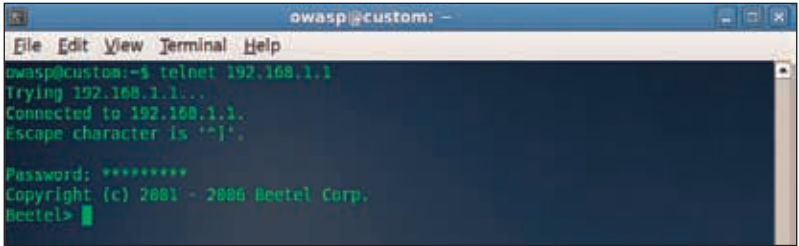
3.1.4 Enumeration / Banner Grabbing

By this time we know the systems we can attack, the operating systems of these systems, the ports open on these systems and perhaps the services running on these ports. Now in order to construct an attack a hacker will need to know the vulnerabilities in the services and operating system the target systems are running.

Footprinting, scanning and now enumeration, each step is more intrusive than the previous. Footprinting required minimal to no interaction with the target system itself, but instead relied on data about the target system available across the internet. In the scanning stage we actually pinged and



Scan profiles for zenmap



A banner grab from a DSL router

mapped the exposed areas of the network. Now we actually actively connect to the system, in order to gain information about the services we are going to exploit. While we are certainly getting more intrusive, we haven't done any hacking yet. Now we are only connecting to the target system in order to gain even more information, and till now we have used only information which the target system is providing anyway.

Banner Grabbing is to connect to a remote system on an open port – as identified while scanning – in order note the output it produces. Often this output will contain the application name, developer, and version number – all useful information for a hacker.

For example, we find that a remote system has the port 21 open. We connect to the remote system using the telnet tool, and observe the remote computer's response. We find out which version of what FTP server software the remote computer is running. This information can be used to search for exploits in online databases. By utilising exploits in older version

```

owasp@custom: ~
File Edit View Terminal Help
owasp@custom:~$ strobe 192.168.1.1
strobe 1.05 (c) 1995-1999 Julian Assange <proff@iq.org>.
192.168.1.1 21 ftp File Transfer [Control] [96,38P]
-> 228 Beutel FTP version 1.0 ready at Thu May 13 20:49:02 201
0\r\n
192.168.1.1 23 telnet Telnet [112,38P]
-> \255\251\3\255\251\1\r\n
-> Password:
owasp@custom:~$

```

Strobe

```

owasp@custom: ~
File Edit View Terminal Help
owasp@custom:~$ nc -v -w2 -z 192.168.1.1 1-100
192.168.1.1: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.1.1] 80 (www) open
(UNKNOWN) [192.168.1.1] 23 (telnet) open
(UNKNOWN) [192.168.1.1] 21 (ftp) open
owasp@custom:~$

```

netcat

of software, we can possibly gain control of remote computer.

Enumeration is a rather complex topic, since each different service needs to be coaxed in a different way to give up this information. For each different type of service, whether it is a web server, a mail server, or a VNC server, knowing the application name and version can go a long way towards discovering an exploit. Going into such detail is out of the scope of this book!



SuperScan 4.0

To secure a system from the ill effects of enumeration, one of the most important rules is to shut down any unnecessary services. Find out if the services you want to run can have their banners turned off, so that it is difficult for a hacker to gain information about application version number which can be used to find exploits for the system. Make good use of a firewall letting in only the minimum required.

3.1.5 Penetration

Finally, what we have been waiting for! The actual hacking act. This step involves discovering and exploiting flaws in the applications running on the remote system. How do we do that? Well, using the information

we have gathered in the previous steps of course!

Online databases of application exploits are available, so that one you have information about the particular version of the remote service you wish to hack, you can simply look up the application in the online database, and you will get a list of



Milw0rm.com has an updated database of most popular exploits

exploits for that particular version of the application in question. A popular example of such a website is <http://www.milw0rm.com/> which is an online searchable database. It is regularly updated with exploits for the latest applications.

Now it is simple matter of using the exploits from these databases to your advantage. However with little or no knowledge of programming you will be unable to develop your own exploits, or understand how the exploits available in online databases work. This is certainly not the ideal situation. You will eventually need to develop some programming skills.

Depending on the severity of the vulnerabilities on the remote system software, and kind of exploit available, you could simply disrupt the functioning of the target computer, or you could end up with shell access to the remote computer as a super-user.

Often exploits will only get you to the point of giving you user-level or guest access to the target system, and after that point, you will have to exploit additional vulnerabilities to escalate your privileges and gain root access.

For now however, let us look at something within scope. We will look at a few exploits that can be exploited by us in our current state. Let us assume that we have already performed the previous steps of footprinting, scanning and banner grabbing, let us move forward from there into the actual exploit.

Let us say that we find out our target is running Kolibri+ 2 Web server on a Windows system. Now we search for this vulnerability in the milw0rm database looking to find an exploit we can use. In this case for example, we find an exploit which matches our needs: <http://www.milw0rm.com/exploits/9650>. Here information about the exploit taken directly from milw0rm:

```
#####  
Kolibri+ Web Server 2 Remote Arbitrary Source Code  
Disclosure aka: More fun with  
Kolibri+ 2 webserver  
Found By: Dr_IDE
```

Tested On: Windows XPSP3

```
#####
```

- Description -

Kolibri+ 2 Web Server is a Windows based HTTP server. This is the latest version of the application available.

This vulnerability is similar to the one reported earlier by Skull-HacKeR.

Kolibri+ 2 is vulnerable to remote arbitrary source code disclosure (download in this case) by the following means.

- Technical Details -

```
http://[ webserver IP]/[ file ][::$DATA]
```

```
http://172.16.2.101/default.asp::$DATA
```

```
http://172.16.2.101/index.php::$DATA
```

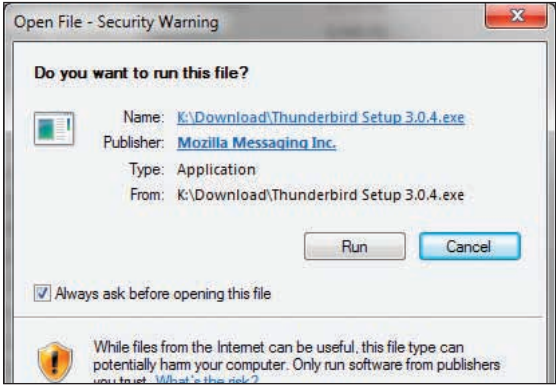
```
# milw0rm.com [2009-09-11]
```

This shows that Kolibri+ Web Server 2 has a serious flaw which makes it disclose the source code of the applications running on it if “::\$DATA” is appended to the end of the file being requested.

This vulnerability exists due to the improper handling of alternate data streams by this application. In Windows systems, for NTFS formatted volumes Windows allows storing more than one data stream for each file. The alternate data streams can be used for a variety of purposes, from storing metadata for a file, storing an album cover for an MP3 for example. When any application accesses a file, it is only able to access the data from the default stream; however, an application that is aware of alternate streams can

access all the streams of a file.

You may have noticed how applications that have been downloaded from the internet display a warning dialog on launching. The dialog may look something like the following:



Ads launch security warning

The reason this happens is that Windows applications that download files from an external server add metadata to the downloaded file that tells Windows that it has been downloaded from a third party. On finding this stream, Windows warns you of the source of the file before launching it. This metadata is stored in an alternate data stream of the file called “Zone.Identifier” which can be accessed as filename.exe:Zone.Identifier. You can see this alternate stream being displayed by the directory listing using the `dir /r` command:



Alternate data streams of files being listed by the “dir” command.

You can inspect this data stream by opening it in notepad from the command line. Just type `notepad filename.extension:Zone.Identifier`.

Back to our exploit. Each file contains one default data stream that contains the actual contents of the file, the ones we see when we double click and open it, or execute it. This stream is called “\$DATA”.

When we access a file hosted on a Kolibri+ Web Server 2 after appending it with `::$DATA` the web server will dutifully fetch this file and return it to you. Normally if you had a PHP or ASP file, the server would execute the code before sending it to the user requesting it. If you are fetching a PHP or ASP file as given in the exploit example:

`http://172.16.2.101/index.php::$DATA`

The server will no longer process the code, as it sees the extension as

.php : \$DATA which it is not configured to process.

Another similar exploit that can affect Windows systems is due to the case-insensitive file system of Windows. Apache on Windows itself is case sensitive, being focused on *NIX platforms. However, it is running on a case-insensitive system.

This vulnerability, called the “CGI Script Source Code Disclosure Vulnerability in Apache for Windows”, retrieved from <http://marc.info/?l=bugtraq&m=115527423727441&w=2> is explained as follows:

ADVISORY NAME:

CGI Script Source Code Disclosure Vulnerability in Apache for Windows

VULNERABLE SYSTEMS:

The vulnerability has been verified on Apache 2.2.2 running on Microsoft Windows XP, Version 2002, Service Pack 2.

FOUND BY:

Susam Pal

FOUND ON:

8th August, 2007

VULNERABILITY TYPE:

Information Disclosure

SYSTEM DESCRIPTION:

Apache HTTPD is a web server that can run on many platforms to provide web-service. The basic server configuration is controlled by the file 'httpd.conf'. The 'DocumentRoot' directive controls which directory is considered to be root for serving documents. For instance:-

```
DocumentRoot "/home/webmaster/site/docroot/"
```

In the above example, a request to 'http://[target]/foo.html' would fetch the 'foo.html' page from '/home/webmaster/site/docroot/' directory of the server.

The 'ScriptAlias' directive controls which directory contains server scripts. The following is an example of a typical 'ScriptAlias' directive:-

```
ScriptAlias /cgi-bin/ "/home/webmaster/site/docroot/cgi-bin"
```

If a user makes a direct request to 'http://[target]/cgi-bin/foo' where 'cgi-

bin' is the scripts' directory and 'foo' is the script, the user gets the output of the 'foo' script. In a secure system, the user is not supposed to view the source-code of 'foo' by making an HTTP GET request.

Vulnerability description

Usually the following directives in 'httpd.conf' file can be considered safe for Unix/Linux (assuming that other directives haven't been insanely edited):

```
# Sample Safe Configuration for Unix/Linux
DocumentRoot "/home/webmaster/site/docroot/"
ScriptAlias /cgi-bin/ "/home/webmaster/site/docroot/cgi-bin"
```

But a similar configuration isn't safe in Windows. For instance:-

```
# Sample Unsafe Configuration for Windows
DocumentRoot "C:/Documents and Settings/webmaster/site/docroot"
ScriptAlias /cgi-bin/ "C:/Documents and Settings/webmaster/site/
docroot/cgi-bin/"
```

If the scripts' directory (represented by 'ScriptAlias') lies inside the document-root directory (represented by 'DocumentRoot') and the name of the script-alias is same as that of the directory containing the scripts then the attacker can obtain the source code of the CGI scripts by making a direct request to **http://[target]/CGI-BIN/foo**.

Apache web-server checks for the exact case mentioned in the 'ScriptAlias' directive before deciding whether the directory mentioned in the Http Get request is a scripts' directory or not. So, when Apache web server receives a request for a file in 'cgi-bin' directory, it finds it to be different from 'cgi-bin' mentioned in the 'ScriptAlias' directive. So, it concludes that it is not a script-alias. Then it checks for 'cgi-bin' directory in the document-root directory and finds it since file-names and directory-names are not case-sensitive on Windows. So, it simply sends the content of the 'foo' file as the HTTP response. It doesn't execute the 'foo' script because it isn't found in a directory pointed by script-alias.

Exploit

The vulnerability can be exploited by making a direct request to **http://[target]/CGI-BIN/foo**.

Prevention

1. Choosing a name for the 'ScriptAlias' different from the name of the actual directory will reduce the risk. For instance,

Sample Configuration for Reducing Risk

```
DocumentRoot "C:/Documents and Settings/webmaster/site/docroot"  
ScriptAlias /cgi-bin/ "C:/Documents and Settings/webmaster/site/  
docroot/sdy1x9y/"
```

The attacker can still get the source code by making a direct request to 'http://[target]/sdy1x9y/foo' if the attacker can somehow determine that the 'ScriptAlias /cgi-bin/' refers to the 'sdy1x9y' directory.

2. A more secure preventive measure would be to place the scripts folder outside the 'DocumentRoot' directory and then form a 'ScriptAlias' to it. For instance,

Sample Configuration for Increased Security

```
DocumentRoot "C:/Documents and Settings/webmaster/site/docroot"  
ScriptAlias /cgi-bin/ "C:/Documents and Settings/webmaster/site/cgi-  
bin"
```

DISCLAIMER:

The information, codes and exploits in this advisory should be used for research, experimentation, bug-fixes and patch-releases only. The author shall not be liable in any event of any damages, incidental or consequential, in connection with, or arising out of this advisory, or its codes and exploits.

Contact information

For more information, please contact:

Susam Pal
Infosys Technologies Ltd.
Survey No. 210, Manikonda Village
Lingampally, Rangareddy District
Hyderabad, PIN 500019
India
Phone No.: +91-9985259521
Email: susam.pal@gmail.com
<http://susampal.blogspot.com/>
<http://securecoding.blogspot.com/>

In short, when a request is made to scripts in the "cgi-bin" directory as **http://[target]/cgi-bin/foo**, the server executes them and returns the result. However if the same script is called as **http://[target]/CGI-BIN/foo** the server no longer executes it, and instead returns the code itself.

The examples we have given above are possibly the simplest attacks that can be made to a server. Most exploits publicly available on websites such as milw0rm are actually available as Perl / Python / PHP / C++ etc., and might sometimes contain intentionally introduced but obvious flaws, in order to discourage usage by people who just wish to download and run such exploits for fun without any knowledge of what they are doing.

Even for these simple attacks, you needed to know about some core OS features such as alternate data streams. You should be able to understand now that it is not a simple task to create such exploits – and these are some of the simplest ones.

In any case, there is a big world out there with some people just looking to have some fun at your expense. With this knowledge of the vulnerabilities your infrastructure faces, you can begin to be prepared. There is no perfect defence, however one must do all they can. Always keep the internet-facing computers updated with the latest patches, to ensure that there is minimal security risk.

While it is impossible to impart a thorough knowledge of penetration testing in such a short book, we hope that what you have now is a good foundation to start understanding some hacking concepts.

3.2 Web application hacking

3.2.1 Introduction

In the previous section, we covered how we could hack network services and possibly gain access to the computers, or its data. Now we will talk about the vulnerabilities of web applications, and how they can be exploited.

The internet is accessible to everyone, and connects everyone. However, it also makes governing it nearly impossible. You never know where your next attack is coming from, and often you might not even know about the vulnerabilities of a system until it is attacked.

The applications you run on your server are accessible to everyone, and can be attacked by everyone. In such a distributed medium, it is simple to find vulnerabilities and easier still to make them public. Even as a vulnerability gets patched, the updates slowly trickle to all web servers using the application. Odds are you will find many are still using older version with the vulnerabilities unpatched. While this might not be of much use to people who already have a particular target in mind, there are many who simply looking for an easy hack without concern of who it is affecting.

That said it is important that the applications you run do not expose such vulnerabilities, because those who attack you are not necessarily your enemies or those who stand to gain something from the attack, but include those who are doing so just for fun.

An increasing number of websites are becoming dynamic. Few websites

today will be simply hosted interlinked html files. Most websites today are dynamic applications that utilize databases and programming scripts that dynamically generate the contents you see on any webpage, and they have been doing so for quite some time. However now an increasing number of websites feature code which runs on the client computer as well. As browsers get faster, the web applications are getting more powerful as well.

Additionally websites are relying increasingly on content that the users themselves contribute. Combined with the dynamic server-side and client-side environments, this exposes a large number of security concerns as malicious users have more vectors of attack.

We shall be looking at some common flaws in web applications that allow web application hackers to wreck varying amounts of havoc. Web application authors are well aware of such vulnerabilities, and as they are found they get patched, however with the growing complexity of web applications mistakes are bound to occur.

The Open Web Application Security Project (OWASP) is a non-profit organization that aims to improve the security of software. To aid this effort, they provide many tools, and documentation for the same. We will be looking at the “OWASP Top 10 for 2010” list, which is their list of the top ten security risks affecting web applications. To demonstrate these attacks we will be using WebGoat, and intentionally insecure web application that the organization has made available. WebGoat – a pun on scapegoat – intends to showcase these vulnerabilities and expects you to attack them to learn more.

We will take a look at each of the ten vulnerabilities from the list, of which a key few will be explored in greater detail. The text within quotes is taken directly from the original report with no “hacking”.

3.2.2 Injection

“Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.”

What this means is that in any application which processes data supplied by the user in an interpreter, care must be taken to ensure that an attacker cannot trick the interpreter into running any code they he / she wants.

This interpreter could be anything, however usually and commonly we hear of SQL Injection attacks, in which attackers take advantage of any part of an application that passes on user-supplied data unprocessed – or poorly processed – to the SQL database.

Due to the ubiquity of SQL servers, we shall look exclusively at SQL Injection attacks, however they are not the sole target of Injection attacks, other interpreters such as XPath, LDAP, etc. can also be exposed the same

way. So how does SQL Injection work?

Imagine a web application that constructs a database query using a parameter supplied by the user. Such a scenario is very common; in fact it is the basis of dynamic websites. So if you were looking at your own profile page on a website, in the backend it would be constructing a query such as:

```
SELECT * FROM user_data WHERE user_id = 32
```

One might expect the URL to read something like:

```
http://www.somesocialsite.com/profile.php?userid=32
```

Now depending on how this query is constructed, this may or may not be susceptible to injection. Of course, we will examine the case where it is. An application susceptible to SQL injection, the code of the application would be something like the following:

```
query = "SELECT * FROM user_data WHERE user_id = " +  
getUrlParameter("userid")
```

This language is of course entirely made up, here is how it would be in PHP:

```
$query = "SELECT * FROM user_data WHERE user_id = ".$_  
GET["userid"];
```

A little about PHP to help you understand the example better: The `$_GET[]` array in PHP is an array which holds all the parameters passed to an application via the URL. Similarly the `$_POST[]` array holds all the POST parameters, i.e. the data passed to the page from web forms. In PHP all variables start with a `$` symbol. The `."` operation joins two strings.

Back to the code. What we see here is that the parameter being supplied via the URL is directly being used to construct the SQL query. Now if we were to add SQL code to the parameter "userid" in the request URL, we could get the website to execute that code.

For example, here we could do something like, **`http://www.somesocialsite.com/profile.php?userid=32;DROP important_table.`**

Since the parameter we provide will be sent to the SQL interpreter unprocessed, we will be able to run any SQL operation we want on the database! Here for example we are dropping – in SQL that means deleting – the table `important_table`. Not a very creative name, but you get the point! With this small vulnerability, the whole database is lost.

One might of course be running the SQL server with a user account that does not have the privileges for this sort of thing, however there are host of attacks which can still be used. You could for example, use this SQL attack to increase your privileges on the site, or to access the list of password hashes, or email accounts for all users.

Let us take an example from WebGoat. In this example, we will try to modify the salary of the user `jsmith` from 50,000 to 200,000. What we are provided with is a textbox that allows us to look up the salary for any user.

Now in this search box, instead of entering a username, we enter:

```
jsmith';UPDATE salaries SET salary=200000 WHERE  
userid='jsmith
```

What this bit of code will do is, to sneak in the second UPDATE SQL command into the original request, and have it execute on the server. This command updates the salary field of the salary table in the database, and sets it to 200,000 instead of 50,000, giving our friend “jsmith” a good 400% boost!

The form below allows a user to view salaries associated with a userid (from the table named **salaries**). This form is vulnerable to String SQL Injection. In order to pass this lesson, use SQL Injection to modify the salary for user **jsmith**.

Enter your userid:

USERID	SALARY
jsmith	50000

The original page.

The form below allows a user to view salaries associated with a userid (from the table named **salaries**). This form is vulnerable to String SQL Injection. In order to pass this lesson, use SQL Injection to modify the salary for user **jsmith**.

Enter your userid:

USERID	SALARY
jsmith	200000

After the injection hack

The way to protect against such attacks is to use a parameterized interface or an application framework API for the database. Each parameter used should be sanitized depending on its type. For example, in the first case we expect a number for the userid, so the parameter should be converted to an integer in the backend, discarding the rest of the string. Strings should be escaped.

We hope that you are now very clear about how an injection attack works, and how important it is to prevent it. With a little thought put in while developing an application it is quite easy to avoid injection attacks.

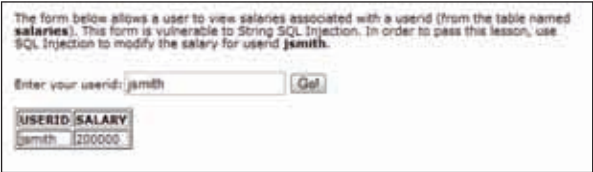
3.2.3 Cross-Site Scripting

“XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.”

Much like the injection vulnerability we explored previously, XSS flaws occur due to improper sanitization of untrusted data. Untrusted data here is any kind of data which does not originate from the server, but is provided in the URL and can be modified by the end-user / attacker.

Unlike the injection vulnerabilities, where the data being run by an interpreter on the server end, an XSS attack is targeted towards the users

visiting a page. An attacker utilized flaws in the coding of the application in order to inject JavaScript code into the web page which can then be used for malicious purposes.



Any part of an application where any input from a URL request is somehow being outputted in the HTML could be vulnerable to such an attack. This kind of attack is of three types, Stored, Reflected and DO-based. We shall cover the first two in detail.

A stored XSS attack relies on vulnerabilities in commenting systems, forums etc where the injected code will persist in a database. For example, in a comment box one might be permitted to post HTML code directly, and while this is rare, such a system could easily be exploited by XSS.

Let us say, one user posts a message which includes some kind of JavaScript element. Since this will posted as a comment, and be part of the webpage, the JavaScript code included will be run on every system which accesses the page with that comment on it.

Since such a script would be running in the context of the end user, it could be used to post personal data about the user to his website.



Let us take an example from the WebGoat. What we have here is a basic messaging system which

We then submit this code to the server, and it shows up in the message list

allows one to post messages which all can read. In our first attack, we will simply create a message which includes a script which displays an alert: `<script>document.location="http://www.maliciouswebsite.com";</script>`

With this script, any user clicking on that link would redirect the

user to “http://www.maliciouswebsite.com” instead of displaying the message.

The second kind of XSS attack is a Reflected XSS Attack. Here the value isn’t stored, but it “reflected” by the target server. To clarify, what reflected here means is that the attack code while being provided by the user is not being processed properly and is somehow ending up on the page.

If there is any way for you to specify a parameter to an webpage via the URL such that the supplied parameter gets injected or “written” to the page directly without being processed, then you have yourself a good old Reflected XSS Attack.

Since the parameter you specify is being written to the page somewhere, you can know that you will be able to run any arbitrary code by providing it as a parameter in the URL. Additionally since the code will appear to be from the trusted site itself, the browser will not be able to block it either.

A hacker could craft a URL which includes the code he wishes to run on the user-end, and distribute this URL through forums / email / comments

Always a good practice to validate all input on the server side. XSS can occur when untrusted user input is used in an HTTP response. In a reflected XSS attack, an attacker crafts a URL with the attack script and post it to another website, email it, or otherwise get him to click on it.

oops! You entered 1111 instead of your three digit code. Please try again.

Shopping Cart			
Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tiling Surface - Cherry	69.99	1	\$69.99
Dynex - Traditional Notebook Case	27.99	1	\$27.99
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$1599.99
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$299.99

The total charged to your credit card: \$1997.96

Enter your credit card number:

Enter your three digit access code:

On clicking this message in the message list, the browser will execute the JavaScript code we entered earlier

oops! You entered injected html instead of your three digit code. Please try again.

Shopping Cart			
Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tiling Surface - Cherry	69.99	1	\$69.99
Dynex - Traditional Notebook Case	27.99	1	\$27.99
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$1599.99
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$299.99

The total charged to your credit card: \$1997.96

Enter your credit card number:

Enter your three digit access code:

In this example we have just displayed a popup, but we could as easily add code to steal session information, or as we can see in the following example, to redirect the user to a malicious website.

etc. Any user who then clicks on such a link is then attacked.

Taking an example from WebGoat. What we have here is a small goods purchase form, the standard deal, which allows you to set quantities, update your cart and purchase using your credit card.

On submitting this form with an “invalid” three digit code we see the following:

As we can see, the invalid value we entered is being directly outputted to the page. Let us take this a step further, and enter some HTML code in the box. We enter a text inside a H1 tag so it will immediately be visible.

We can include a script tag in the input box too, and the page will inject then in the error message, thus executing it.

Now that we know this field is vulnerable, what do we do? We certainly cannot ask users to input our code in a text box!

By inspecting the page, we see that the “three digit access code” field has a name of “field1” in the form. If we could specify a value for that in the URL, we wouldn’t need the user to enter it, as it would be pre-populated, and

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	<input type="text" value="1"/>	\$69.99
Dynex - Traditional Notebook Case	27.99	<input type="text" value="1"/>	\$27.99
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	<input type="text" value="1"/>	\$1599.99
3 - Year Performance Service Plan \$2000 and Over	299.99	<input type="text" value="1"/>	\$299.99

The total charged to your credit card: \$1997.96

Update Cart

Enter your credit card number:

Enter your three digit access code:

Purchase

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

* Whoops! You entered 1111 instead of your three digit code. Please try again.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	<input type="text" value="1"/>	\$69.99
Dynex - Traditional Notebook Case	27.99	<input type="text" value="1"/>	\$27.99
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	<input type="text" value="1"/>	\$1599.99
3 - Year Performance Service Plan \$1000 and Over	299.99	<input type="text" value="1"/>	\$299.99

The total charged to your credit card: \$1997.96

Update Cart

Enter your credit card number:

Enter your three digit access code:

Purchase

would directly launch the attack.

So we construct the following URL:

http://localhost/WebGoat/attack?Screen=191&menu=900&field1=<script>alert("Gotcha!");</script>

This pre-injects the script into the page, and any user who follows this link will automatically run the script specified in the URL.

Note: The URL given above is for an instance of WebGoat running on the local computer. After downloading, installing and launching WebGoat, you will be able to test it as well.

The third DOM-based XSS Attacks rely on a web page's use of the document.url or document.location properties. For example, a web page that is displaying all entries for a tag might have a URL such as:

http://www.somesite.com/search.php?tag=android

This webpage might be retrieving the url, and extracting the tag name from the parameters list in order to display it on the page. So we could use a URL such as:

http://www.somesite.com/search.php?tag=<script>alert("Gotcha!");</script>

Then the webpage just needs to process the URL, and the DOM will do the rest.

* Whoops! You entered injected html instead of your three digit code. Please try again.

Shopping Cart			
Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tiling Surface - Cherry	\$69.99	1	\$69.99
Dynex - Traditional Notebook Case	\$27.99	1	\$27.99
Hewlett-Packard - Pavilion Notebook with Intel Centrino	\$199.99	1	\$199.99
3 - Year Performance Service Plan \$1000 and Over	\$299.99	1	\$299.99

The total charged to your credit card: \$1997.96

Enter your credit card number:

Enter your three digit access code:

3.2.4 Broken authentication and session management

“Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users’ identities.”

Such vulnerabilities exist because web site creators are not careful enough about the security of their authentication and session management. Websites need to be built keeping in mind that user’s may not always be as careful with the security of their accounts.

Users don’t always log out of their accounts when they are done, and they are often not careful about the data they might be exposing by sharing

private links with others. Often such risks are not clear to the end user, and more often than not they shouldn't be there in the first place. For example, a user browsing a shopping site might assign a cart id under which all your items are added. The user then shares a link to his cart page with a friend who might be interested the same. Since this link uses includes the cart id, the friend is unknowingly using the same cart, and when he purchases from that cart, the money is deducted from the original users account.

This is a huge flaw in the system, and might be more common in web sites that allow a user to shop without needing to register.

A malicious user could use such links that include the session id or cart id to hijack the identity of the original user and conduct transactions on their behalf. If such session ids are not random enough, an attacker might be able to guess the session ids of others, or even brute-force their way to a valid session id and then take on the identity of such a user. Sessions should be timed out appropriately, so that a user who has simply closed the browser window without logging out first is protected.

While both the parameters sent via GET in the URL and those POSTed can be inspected by an attacker, GET parameters occur in the URL and will be stored in the history in the browser, and in the cache. This information can be invaluable to an attacker.

Error messages that a web site returns can also be a source of attack. While it may seem helpful for you to give the information as much information as possible, this information can also be used by an attacker. Any sensitive information that establishes the identity of a user should be transmitted over an encrypted connection.

For example, when you enter an incorrect password, the application specifies that while the login id is correct, the password is not. What is wrong with this picture? Well, for one you have just exposed that the user has an account on this website! This is why most websites will serve a message like "Invalid user ID or password." It is not enough that the message that the user can see does not expose anything, but nothing in the URL, or the source of the document or the request structure should betray this information either. If an attacker can establish that the website takes more time to process an invalid password than it takes to process an invalid username, then that is information enough for the attacker.

For an attacker who already has a login, the process of breaking the password should not be simple. If a website allows a user indefinite tries for logging in, then it is vulnerable to a password cracking attack. It is not only important that the login page be secure from such attacks, but also the "forgot password" page. If an attacker has indefinite tries to guess a user's password, then eventually he will simply be able to guess the right values.

Also important is that you store all passwords as hashed values – salted

as well – in the database. If the passwords are stored in the database as plain text, an attack on the database will lose you all your user account passwords. A simple way to test if a website is storing passwords as plain text is to create an account, and use the forgot password feature. If the website is able to tell you your original password via email or on the site itself, they certainly have it stored somewhere or the other.

Nearly any website of note today uses some measure of session management and authentication, to provide users with a more dynamic environment. However, it is their responsibility to ensure that their website is secure from such attacks even if the website does not have much private information.

3.2.5 Insecure direct object references

“A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorised data.”

Often in order to simplify the working of a web site, the developers will directly expose many internal objects of the system. Such objects while not intended to be directly manipulated by the user could be altered by an attacker to their advantage.

To understand this more clearly, let us construct an imaginary messaging system, like that found in forums, or one like the direct messages feature of Twitter. In this application, each user has an email id, and can post messages to any email id on their friend list. Seems real enough?

Our hypothetical messaging system posts messages to users by sending them to a particular URL, which includes the user's email id and the message to post. For example:

`http://www.vmessage.com/postmessage.php?from=someone@someplace.com&to=abcef@xyz.com&message=hey what's up`

The developers of the `postmessage.php` script decided it would be simplest to just pass all the information in the URL itself and keep the script simple. Instead of checking whether the target user even belongs to the contact list of the logged in user, it assumes that the frontend would only be using this URL for emails in a person's contact list. For most users this might be the case, however for an attacker this is an excellent opportunity.

The application should validate the contact id in the database, and instead of using the email address directly, it should have used a more cryptic contacted which would only be valid for the current user. This could leave the URL as:

`http://www.vmessage.com/postmessage.php? to=63&message=hey what's up`

If the contact id here refers to a contact in the own users list, then the most

a hacker could do is post a message to someone in the own contact list using an id.

The idea here is to expose as little as possible about the internal working of the system. If there are any obvious parameters that people are able to manipulate, don't expect them not to.

Let us look at a famous example of such an exploit in Hotmail, back in 2001. A user accessing their email account could see a URL such as:

<http://lw2fd.hotmail.msn.com/cgi-bin/getmsg?curmbox=F000000001&a=5691b2b44e104176111971aa0fbb1274&msg=MSG998000947.3&start=197078&len=1060&msgread=1&mfs=182>

Here the URL is exposing an internal data object, the message ID, which here is (MSG998000947.3). As it so happens, this message id is unique for each message. Using this message ID, one could access this message even when they are not logged into the account with which this email is associated.

So if the message (MSG998000947.3) existed for the user hackedguy@hotmail.com, some other Hotmail user haxOr@hotmail.com could access this message from their own account with a specially crafter URL. A hacker would just need to insert the right values into the right locations in the attach URL in order to see the message:

http://pv2fd.pav2.hotmail.msn.com/cgi-bin/saferd?_lang=EN&hm__tg=http%3a%2f%2f64%2e4%2e36%2e250%2fcgi%2dbin%2fgetmsg&hm__qs=%26msg%3dMSGXXXXXXXXX%2eX%26start%3d1%26len%3d9999999999%26login%3dUSERNAME%26domain%3dhotmail%2ecom

Note: The unescaped URL is: http://pv2fd.pav2.hotmail.msn.com/cgi-bin/saferd?_lang=EN&hm__tg=http://64.4.36.250/cgi-bin/getmsg&hm__qs=&msg=MSGXXXXXXXXX.X&start=1&len=9999999999&login=USERNAME&domain=hottmail.com

Here the attacker would enter the message number (MSG998000947) in the bolded location marked MSGXXXXXXXXXX, and the value after the period (3) in the original URL in the X following that. In the place marked USERNAME, the attacker would place the username of the user to whom the message belongs. Finally, we would have a URL as follows:

http://pv2fd.pav2.hotmail.msn.com/cgi-bin/saferd?_lang=EN&hm__tg=http%3a%2f%2f64%2e4%2e36%2e250%2fcgi%2dbin%2fgetmsg&hm__qs=%26msg%MSG998000947%2e3%26start%3d1%26len%3d9999999999%26login%3dhackedguy%26domain%3dhotmail%2ecom

On visiting this URL in the browser while being logged into any other Hotmail account, an attacker could view messages belonging to another user. People could brute force the message codes by trying different values until one worked and get access to other people's emails. Hacking utilities for finding valid message codes also started becoming available.

3.2.6 Cross Site Request Forgery (CSRF)

“A CSRF attack forces a logged-on victim’s browser to send a forged HTTP request, including the victim’s session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim’s browser to generate requests the vulnerable application thinks are legitimate requests from the victim.”

For most web attacks authentication is important. The attacker must disguise the attack as if it is an authenticated / authorized request. The genuineness of the attack makes it difficult to detect and block.

What if then, you could have the end-user himself do your dirty deed for you. Instead of trying to impersonate a user, or bypass the authentication system, how about using the user to conduct the attack on himself / herself?

In simple words, why bother to forge a signature when you can forge the amount with greater ease, when you have a blank check ready-signed.

A Cross-site request forgery does exactly that. Instead of forging the user credentials, you forge the request that will get you what you need, and somehow have the user perform the request. What CSRF relies on is that any request sent by a logged in user will be considered legitimate, and the web site will have no way of blocking it.

For a transaction conducted on a money transfer website such as PayPal, how is the website to tell if the user really wanted to transfer funds to the hacker, or whether it was an attack if it is coming from a legitimate authorized user. After all, transferring funds is a feature of the website, and they expect such usage from their users.

You might be wondering how exactly such an attack would be constructed. What we have explained till now doesn’t really cover it. Despite the seeming ambiguity of such attacks, they are actually quite simple. Don’t worry; the following example will make it very clear:

Let us take an example of an online multi-player strategy game. The game – as usual for its genre – involves gathering resources such as wood and gold. Now suppose that an attacker discovers that when he transfers gold to another player in the game, the URL request looks like the following:

`http://awesome.strategygame.com/transfer.php?item=gold&user=sumguy&amount=10000`

Now he can’t change the sender of the money, only the person who will receive the money, and the amount. As such changing the URL will not result in the user being able to do anything they cannot already do with the interface provided. What they need to do is to somehow get other users to visit this URL while logged in, and have it transfer money to the hackers account.

What the hacker now does is posts some specially constructed HTML code in his profile page, or in the forum section for the game, or anywhere

else actually where he would expect gamers of this game to visit. Here is what he might post:

```

```

Can you imagine what will happen?

Every time anyone visits a page with this image embedded in it, his browser will fire off a request to the URL:

`http://awesome.strategygame.com/transfer.php?item=gold&user=hax0r&amount=10000`

This URL will try to transfer funds to our friend “hax0r” from the account of such a user. While this URL does not resolve to an image, the browser will send off a request anyway, and since it is a 1 px x 1 px image the user will not see anything.

There you have it, we got a user to without knowing it, hand over his gold to you! For this they needed to do nothing more than just visit any webpage which had this image in it. Of course they needed to be logged into the game first; however for visitors to the hacker’s profile page at the gaming site, or the gaming forums, this will be highly likely.

The web site has no way of distinguishing such a request from a genuine request, and as such they cannot block it as is. However, that does not mean there is no way to stop such attacks.

Disallowing images in the forum or in the profile page might not be what is desired. A second confirmation page is also not really a solution and might hinder the experience of the website. An example of how to hack a website that uses a confirmation page is as follows:

```
 <img id="image2" width="1" height="1" />
```

In the example given above, the first image requests a gold transfer, and when that fails – and it fails because the URL is not a valid image – it sends the confirmation using a second image. These attacks could also have been conducted via iframes.

There is a way out though, a way to protect your web application and your users from such attacks. Such attacks can be avoided by the use of a unique session or request token. Since this token would not be something that an attacker will know, and because the token will change frequently such attacks will no longer be possible.

3.2.7 Security misconfiguration

“Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained, as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.”

This one is pretty straightforward; however it encompasses a wide range of flaws. Getting your website or web application to work is important enough, however enough time needs to be spent on its security.

Nowadays it is pretty simple to set up a basic website. There is no dearth of free and commercial content management systems, blogging platforms, social platforms, forums etc. On the client side as well, we see a great many frameworks that aim to make a web site richer and more interactive with the use of technologies such as Ajax.

However, these applications themselves are often vulnerable, and susceptible to attack. Furthermore, many of these applications ship with default settings that are intended to work for the majority of people and might not be secure. A very serious oversight would be to not change the default password which ships with the system. An attacker who tries the defaults could then access your system with administrative privileges.

From time to time authors of these applications release updates that address these issues, however most people do not update as regularly. This means that often what websites are running are older vulnerable versions.

A misconfiguration of the web server that allows directory listing could expose important information about your site, and might help attackers discover flaws in your system in order to construct future attacks.

Often such tools also come with an administrative or wizard interface which is intended for one-time usage to help set up the server. Many times the default settings enable debug traces on the front end to help you debug. While all this might be convenient for you while setting up and testing the website, these can be used by hackers to construct attacks.

Security misconfigurations are an important source of vulnerabilities as they expose and prolong flaws that have been fixed and give hackers access to interfaces that were only intended for you.

3.2.8 Insecure cryptographic storage

“Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.”

When any kind of private data belonging to a user is stored on a web server, it is the host's responsibility to ensure that such data is not exposed to

anyone other than the user himself. To ensure this they encrypt all sensitive user data.

However all data will eventually need to be decrypted in order to be used by a user of the site. Improper access control measures can lead to the leakage of such data if the mechanism used to store and retrieve them is not proper.

Imagine a hypothetical system where credit card numbers are stored in a table for use by a web application. To ensure that the credit card numbers will be secure in case of a database leak, the values are stored in encrypted form in the database. However, the application is set to decrypt these entries automatically when requested by the server. In this case, an attacker could use a SQL injection vulnerability to expose the credit card numbers.

As you will learn in the section about passwords and hashes, an unsalted hash is vulnerable to decryption. If the password for the users is being stored as an unsalted hash, accidental exposure of these passwords via a vulnerability will be much more harmful.

Another example – which parallels writing your computers administrator password on a sticky note on you monitor – is to actually leave your password exposed in the same volume as an encrypted backup! To save from remembering yet another password, the person conducting the backup might store the password in; say the volume label of the DVD with the backup on it. It is clear that simply using encryption is not enough, the decryption procedure, and key need to be kept secure as well.

3.2.9 Failure to restrict URL access

“Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.”

Most websites will be careful about restricting the entry points of their administrative or restricted interfaces. So a website will likely restrict access to URLs for their administrative interfaces with login pages, but what about what lies under them?

For example, a web site <http://www.somesite.com> has an administrative section of the site under the “/admin” directory which allows them to add, remove and update content on the website. The administration section is guarded with a password. To make things easier for those working on the site, the owners of **somesite.com** decided to install a file management application to the backend. This application would give an administrator full control of the websites files. Unfortunately, this application is not well integrated with the authentication mechanism of the site, and attackers find that by accessing the URL for the file browser directly at “<http://www.somesite.com/admin/filebrowser>”, without going through the administrative

interface they are able to use the application without privileges.

Another case of such vulnerability might exist if the web site simply checks if the user is logged in, but doesn't check the authorisation level. For example, a CMS might show or hide links to certain features, such as editing articles and uploading files depending on the user's level of access; however simply showing and hiding such links is not enough if the user can simply type the URL and access these features. Each page to check whether the current user is authorised to access it instead of just relying on the absence of links to the interface as a deterrent.

The role of the user using the web site should be considered by the web site instead of a simple binary logged in / logged out check.

3.2.10 Insufficient transport layer protection

"Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly."

As was made clear earlier, simple encryption isn't enough, the encryption itself needs to be secure. However often times, sensitive information being transmitted by the browser is not encrypted at all.

It is remarkable how much information can be tapped by simply using application such as Wireshark to listen in to network traffic. Any and all content which is transmitted without encryption is as good as public knowledge. If any password is being transmitted without SSL encryption to the server, it doesn't matter how well it is hashed in the server, with how much salt, an attacker can simply listen in and get the password – or other secure information – in transit.

A website needs to ensure that any webpage that requires authentication, or is required to transmit private secure data such as credit card numbers is sufficiently secured with a VALID SSL certificate.

3.2.11 Unvalidated redirects and forwards

"Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages."

This is again a pretty straightforward exploit. Consider a web site that uses a redirect script to transfer users to another external site. This gives phishers a considerable boost, as they can use the original domain name to gain some authenticity.

So for a social networking website which has a redirect page:

<http://www.youttwitface.com/redirect.php?url=http://www.totallyevilsite.com>

If such a link is posted in the social networking website itself, a user might consider it safe since it begins with the familiar name of the social networking site. The user on clicking on this link is forwarded to a phishing site `totallyevilsite.com` which has a login dialog for the social networking site `youttwitface.com`. The user – thinking that he is simply being asked to confirm his credentials or re-login due to an expired session – enters his login credentials. The rest is history.

Another example of a similar exploit would be if the website has some forwarding mechanism to transfer users from one section of the website to another. For example if a user needs to confirm his age before entering a mature section of the site, the website after checking his age will use some forwarding mechanism to get the user to where he wanted to go. Such a URL might be as follows:

`http://www.youttwitface.com/ageconfirm.php?forward=adult.php`

What if an attacker changes this to:

`http://www.youttwitface.com/ageconfirm.php?forward=admin.php`

If the forwarding mechanism is poor it might redirect the user to sections of the website to which he / she would not otherwise have access.

3.2.12 Conclusion

The web has never been a safe place, but today with the proliferation of new technologies and web development concepts, hackers have an even greater surface to attack. As we wait for the new HTML5-based web standard to emerge, you can be damn sure they will be exploited in ways we cannot imagine right now.

It helps to be prepared, especially when the attack could come from anywhere. You need to think of each user who visits your website as a potential hacker; your site doesn't get 10,000 visits a month, it gets 10,000 potential attackers. A cynical way to look at things surely, but it is also an effort to protect the interests of your other users, err... potential attackers.

No one can ever be fully secure, there are some very smart hackers out there, and many of them are responsible for discovering these exploits in the first place. It is common that a flaw, which would allow an application to be compromised, is only discovered after the fact.

Be prepared for the worst. 

4 Conclusion



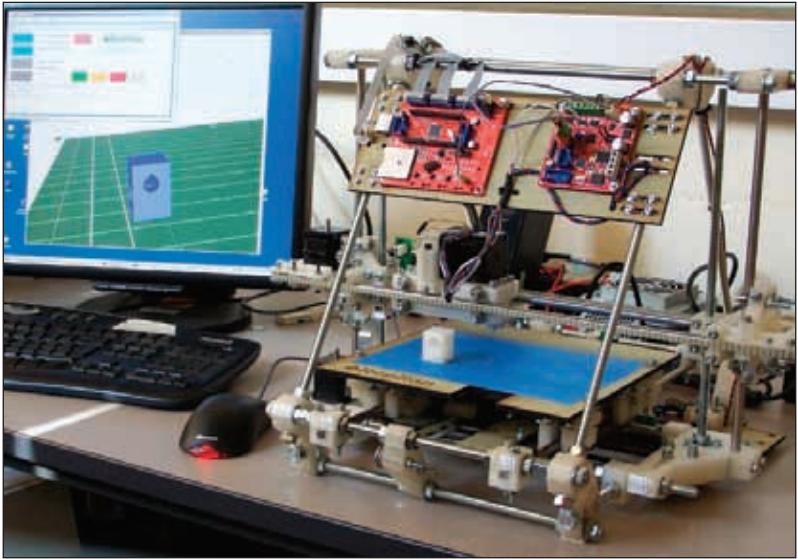
4.1 Hacking... Anything

While we have been focusing on web, the networks and information security in this Fast Track, this is not what hacking is all about. We have tried to make this quite clear in the introduction of this book. After reading this book, you should not limit yourself to thinking of hacking purely in terms of security and computers. There is a whole world out there waiting to be hacked.

While most of the situations we have covered require that the application design be tightened, and support fewer parameters, sometimes applications are better off if they are hackable. Mozilla Firefox is a brilliant example of this.

Nearly every part of the application is configurable; while other browsers are allowing introducing features such as jump lists and tab previews, Firefox is the only browser to allow one to turn these features off, and control how they function – although this particular feature is only in the unreleased versions. Each and every aspect of the Firefox UI can be altered with ease. Chances are if you don't like the way Firefox handles something you will find a setting to turn it off or change it from about : config.

The Firefox UI is described in plain text format with a language called XUL. A large portion of its functionality comes from code written in JavaScript, which is editable by the end user. Firefox is hackable to the core. While exposing this to each and every user would just complicate matters for



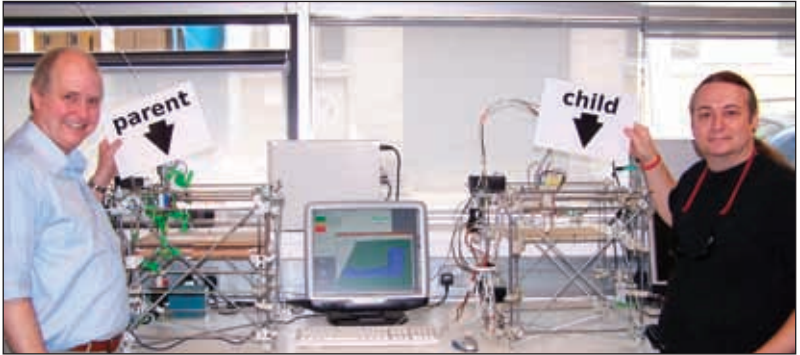
The RepRap connected to a computer.

them, and few would appreciate a settings dialog with thousands of settings, it does make Firefox a more lucrative choice for those who like to have things exactly the way they want. Now Mozilla has also launched a service for creating customized versions of Firefox for distribution within communities.

If there is anything to be learnt from the success of Firefox, it is that hackable applications can be good, and simple to use while still being powerful.

Linux has always been hackable in essence, allowing nearly every component to be replaced and modified to heart's content. After all, the OS is open source, how much more hackable could it get! This is the reason why it is popular amongst hackers and those who like fine-tuned control over their computers. For this reason, Linux has suffered as a desktop platform, as a majority of people would prefer a system which Just Works™, rather than one which Works Just the Way You Want Once You Configure It Through a Hundred Text Files™. Recently Linux has become simple enough that it works out-of-the-box for most people. Hopefully Linux will continue to become easier to use while retaining its power and hackability.

Open Source is the way to go, not just because it promotes freedom, but also because it fosters creativity. You want to know how an application works? Well, you can find out. While trying to find out how a proprietary application works will probably make you a criminal, free – as in speech –



The project founder of RepRap Adrian Bowyer (left), and the software developer Vik Oliver (right) showing a RepRap machine created by another RepRap machine.

software encourage you to find out how they work, and contribute back to the community.

We now see such a culture developing around hardware devices as well. While few will trust an open source hardware device designed by unknown community members, a free flow of information regarding hardware designs simply means that hardware will have the opportunity to innovate as fast as software. One example of a popular open source hardware platform is Arduino. It is a hardware platform which uses an Atmel ATmega168 microcontroller, and is designed to be simple to program and interface with a computer. Arduino also includes a software component – which is also open source – which is an IDE and programming interface for Arduino hardware. Since the specifications for Arduino are open and free, any hardware manufacturer can create Arduino-based boards which will work with the Arduino IDE. If you had any Arduino-based device, you could simply connect it to your computer and reprogram it to do whatever you wanted.

One of the most brilliant projects based on Arduino would have to be the RepRap project (reprap.org). The RepRap (Replicating Rapid-prototyper) device is a 3D printer that has an open design, and is developed with community participation. The firmware, and the actual design of the device are both open, so people are free to make their own RepRap machines and to improve upon them.


One of the most brilliant aspects of RepRap is that it can print many of its own parts! One you have one RepRap machine, you can construct another one using parts available in stores and parts printed with the RepRap machine! What better example could there be of a self-propagating open source hardware device?

While developing this device the engineers found that the specifications of the current Arduino-based boards were not enough for a project of this

scope, and for need to be higher for compatibility with future models. So guess what they did. They hacked it.

They created a new board called Sanguino, which was compatible with the Arduino but featured four times the RAM, four times the flash storage and more I/O pins for controlling other devices. The Sanguino design, like its parent Arduino is free and open source. By using the same IDE they needn't reinvent the wheel when it comes to writing code for the device, and can get the support of the Arduino community.

Imagine instead what the case would have been if the Arduino was not free, if they couldn't hack it. This brilliant and innovative product would have been crippled.

We are approaching a future where people will, hopefully, be better educated about hacking and its benefits. If this society could simply take one step from advocating a Copy-Paste culture to Copy-Improve-Paste, the world will be much better off for it. 

5 Appendices

5.1 Before you begin hacking

There is a sense of satisfaction in getting software to do something its author never intended or perhaps even imagined. Needless to say this also means you are going into untested territory – and while there is a charm in that – it can end up crashing your system and exposing bugs. Before you go messing about with your computer, it is a good idea to have a recovery plan. If you use Windows make sure you have the System Restore / System Protection turned on. Windows XP, Vista and 7 discs include a recovery mode that can be activated in case of any problems.

In Windows XP you can install the recovery console to your hard disk if you choose. This way you can easily access the recovery console without the long loading times when running it from the CD. The following steps will guide you how to install the recovery console on your hard disk:

- Insert the Windows XP disk into your disk drive
- Click on Start > Run or press the [Windows] + [R] to launch the run dialog.
- In the run dialog enter `X:\i386\winnt32.exe /cmdcons` (replace X with your disc drive letter).
- Follow the instructions on screen.

If you are using Windows Vista or 7, you might already have the recovery console installed on the hard disk. In this case you will find a “Repair your computer” option when you press [F8] while starting your computer. If you don’t see this option, you can still boot from the Windows DVD and choose the Repair your computer option. In this case it is better to create a bootable pendrive with the Windows Vista / Windows 7 installer. If you have a Windows Vista or Windows 7 disc, you might prefer to use the recovery console provided with that, as it is more powerful and will come in handy.

To create a Windows Vista / 7 bootable installation pendrive, you will need a 4GB pendrive. Once you have one, follow the steps below. Since this book is aimed to enlighten you as much as possible, we will describe what we are doing here and the command line tools and options even though it is slightly tangential.

- **Connect the pendrive, and format it in NTFS**

This step is important because the Windows Vista or Windows 7 setup can only be booted from an NTFS drive. If you are using Windows XP, you will not see an NTFS option while formatting pendrives. Use a Vista / Windows 7 system or the bootable DVD.

- **Launch the command prompt as administrator by right-clicking on it in the start menu and clicking on Run as Administrator.**

Since the steps we are performing will write directly to the pendrive it needs to be done with Administrator privileges.

- **Type diskpart and press [Enter].**

The command line utility “diskpart” is a powerful disk partitioning utility which is included with Windows. It can be used to perform many advanced partitioning operations. It can be used for creating / removing / shrinking / growing partitions. It can also be used for formatting pendrives in NTFS on Windows XP. However for the steps that follow, the Windows XP version of diskpart will not be sufficient. You can run the command from the recovery mode from a Windows Vista / 7 disc though.

- **Type “list volume” press enter**

This will list the volumes on the system. This will list each and every drive / partition on your computer and will list its volume number, drive letter label etc. Note the volume number for the pendrive you just formatted.

- **Type “select volume <pendrive volume number>” and press enter**

This will select the pendrive volume so that you can perform operations on it.

- **Type “active” and press enter**

This will set the “active” flag on the pendrive volume, without which you cannot boot from it.

- **Exit from diskpart using the “exit” command**

- **Insert the Windows Vista / 7 disc**

- **In the command prompt navigate to the “boot” directory on your DVD.**

- **Type “bootsect.exe /nt60 <pendrive letter>”**

The bootsect utility will transfer the boot sector to the pendrive so that it becomes bootable. The “/nt60” parameter is to specify that the Windows Vista-compatible boot-code needs to be applied.

- **Now copy all the files from the Windows Vista / 7 DVD to the root of the pendrive.**

You now have a bootable pendrive with the Windows Vista or Windows 7 setup. Booting into the recovery mode, and even installing Windows Vista or

7 from the pendrive will be much faster than by DVD.

Besides a Windows disk, it might be a good idea to keep a Linux disc / pen drive handy too. We would recommend BackTrack Linux 4. While it is based on the rather dated Ubuntu 8.10, BackTrack 4 is a Linux distribution designed for penetration testing. You can download a liveDVD of the OS from their web site: <http://www.backtrack-linux.org/>. Using a tool such as Unetbootin, you can easily transfer this disk image to a pendrive and boot off that. Linux can be very useful for debugging Windows problems as well, and often allows you to access your system at a lower level than you could with Windows. However, for full fledged NTFS support, and to manipulate your computers registry offline, it is better to use a Windows recovery disk / pendrive.

5.2 The Windows registry

For all the people using Windows, it is very helpful to learn about one of the most important aspects of any windows system, the Windows Registry. The registry in Windows is a central database of all configuration settings for most of the applications installed on your computer. It is used extensively by Windows for storing all its settings and by most Windows applications. The Windows registry allows one to configure many settings for Windows and other application which are otherwise not accessibly by any interface that the application provides.

By understanding a few things about the Windows registry you can go a long way towards hacking your system into what you want it to be. Editing the Windows registry is one of the most powerful ways of manipulating you system settings, which is something that makes it equally dangerous. It is a minefield filled with often cryptic configuration parameters which usually affect your system in ways you won't even notice. It can completely mess up your system in such a way that you are unable to even boot into Windows. Modify the right hardware setting, and your hardware gets a boost, modify the wrong one, and you end up with a non-booting system.

So before you set out to do anything with the registry, please back up your system registry. If you are using the System Restore / System Protection feature in Windows create a recovery point. Before going into your registry and editing things indiscriminately, read up about what you want to accomplish – Google is your friend here, unless you use Bing :-)) – to get an idea of what you might need to change. Before you edit your registry see if there is any way to change the setting in an way supported by the application. Wait, this is book about hacking right? Disregard that!

Even so, there are many tools out there which will help you accomplish some advanced registry manipulations without needing to edit it manually.

For example, a common use of editing the registry is to remove applications from your startup sequence. You might think that the “msconfig” utility included in all Windows versions might do the trick, however the “msconfig” utility only exposes the tip of the iceberg.

For a better idea of what you put your system through each time you start it, try the “autoruns” utility by Microsoft SysInternals. This utility will list nearly each and every application and dll that the system loads during its boot procedure. It includes not only the applications launched on login, but also components which are loaded by Windows Explorer. Using “autoruns” you can thus remove any unwanted entries which have polluted your right click menu. You can also disable services, disable hardware drivers (might make your system unbootable), disable enabled Windows Vista / 7 gadgets and much much more. While it is certainly possible to do all this straight using the registry, the configuration locations in the registry are too spread out. Another good thing about the “autoruns” utility is that it provides you the registry path for all its settings, so you can use it to find popular registry locations. The Windows registry is like a virtual file system in which you have “keys” and “values” instead of folders and files. A key is reminiscent of a folder in a drive, as it can contain any number of sub-keys (which in turn can host further sub-keys). A key can contain multiple name-data pairs which represent the configuration entries. Each key has a default entry which can also be assigned a value. The data in each name-data pair is typed, i.e. it can only contain data of a certain type. The data types available are String, Binary, DWORD, QWORD, Expandable string and Multi-string. Just like you have with your filesystem, the registry also uses paths to specify the location of any particular key.

To edit the Windows registry, the simplest tool to use -- one which is free and works reasonably well -- is the registry editor which comes bundled with Windows: regedit. It can be launched using the Run dialog (Windows Button + R) by entering the command `regedit` and hitting [Enter].

The Windows registry in any recent OS will display the following 5 root key entries:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG

All these settings are not stored in a single file, but are instead spread across multiple files in logical groups called “hives”. The combination of data from these “hives” makes up the Windows registry. The key to hacking your registry is to understand where everything goes. Let us look at the kind of settings stored in each of these sections to gain a better

idea of how the registry is organised, and where you are likely to find the setting you want to change. Once you know that you will be able to navigate through the registry with much greater ease while looking for the setting you want to change.

- **HKEY_LOCAL_MACHINE**

Abbreviated as HKLM. According to its description in Microsoft TechNet: The HKEY_LOCAL_MACHINE subtree contains information about the local computer system, including hardware and operating system data, such as bus type, system memory, device drivers, and startup control parameters. The Software subkey in this key contains the bulk of the system-wide settings for applications installed in Windows. The settings in this section are usually arranged as Company/Product/Version as those are Microsoft's guidelines. Settings for Microsoft's own applications will thus be found in HKLM/Software/Microsoft, with setting for Windows itself under Windows.

- **HKEY_USERS**

This key contains user configuration keys for each user. It has one subkey for each user which is named after the user's security ID. For an explanation of what is contained in the subkeys read on for HKEY_CURRENT_USER.

- **HKEY_CURRENT_USER**

Abbreviated as HKCU. According to its description in Microsoft TechNet: The HKEY_CURRENT_USER subtree contains the user profile for the user who is currently logged on to the computer. The user profile includes environment variables, personal program groups, desktop settings, network connections, printers, and application preferences. The data in the user profile is similar to the data stored in the Win.ini file in Windows 3. x.

The HKEY_CURRENT_USER subtree does not contain any data. It just stores a pointer to the content of the HKEY_USERS\ Security ID (SID) of current user subkey. Therefore, the content of that subkey also appear in HKEY_CURRENT_USER, and it can be viewed and changed in either location. This subtree provides easier access to the data.

A new HKEY_CURRENT_USER subtree is created each time a user logs on. The data for the subtree comes from the profile of the current user. If no profile is available, the subtree is built from the user profile settings established for a default user, which are stored in System drive \ Documents and Settings\Default User (WINNT)\Ntuser.dat. While HKEY_LOCAL_MACHINE contains settings applicable machine-wide, this registry location is applicable only to the current user. As such it contains entries similar to HKEY_LOCAL_MACHINE.

- **HKEY_CLASSES_ROOT**

Abbreviated as HKCR, this part of the registry is for storing file associations and information about application registered for handling different data types. These settings can be overridden by the current user's settings for the same stored in HKEY_CURRENT_USER/Software/Classes.

- **HKEY_CURRENT_CONFIG**

It merely links to HKEY_LOCAL_MACHINE\System\CurrentControlSet\Hardware Profiles\Current and contains information generated at runtime.

Quite a bit of advanced registry manipulation can also be done using the Microsoft Management Console, which can be launched using the command “mmc”. The Microsoft Management Console can add “snap-in” each of which allows configuring a different part of your computer. Of particular note is the Group Policy Editor which can be launched with the “gpedit.msc” command. This will allow you to configure your computers “policies” such as password expiry and password strength restrictions etc.

Now when you install an applications, you will have some idea about where exactly it stores its settings, and where all those settings reflect. Since the registry is meant to be used internally by an application, often it is not obvious what a setting does, and how it might affect your system, or even what the setting values mean. But you're going to go ahead and change it anyway aren't you. Naughty!

You are unlikely to find any kind of documentation from an application creator about the registry either. Unless the application is open-source, in which case the source itself is documentation, if you can read it! As said before, the registry is a dangerous place to hack around in, be safe and backup.

5.3 Port Lists

5.4 Web Resources

- <http://www.milw0rm.com>

Online database of application exploits

- <http://www.securityfocus.com>

Online database of application exploits

- <http://www.metasploit.com>

Penetration and security testing framework

- <http://www.foundstone.com/>

Tools relating to web and application security by McAfee's

- <http://googleonlinesecurity.blogspot.com/>

Google's web security blog

- <http://www.remote-exploit.org/>


- <http://ophcrack.sourceforge.net/>

A utility for cracking Windows XP and Vista passwords using rainbow tables. Also available as a liveCD distro.

- <http://md5.rednoize.com/>

An online tool for decrypting MD5 and SHA-1 hashes.

- <http://project-rainbowcrack.com/>

One of the best tools for generating rainbow tables, and cracking hashes with them. It works with MD5, SHA-1, windows password hashes and many more. 

Port Lists			
Port	TCP	UDP	Description
0	TCP	UDP	Reserved
1	TCP	UDP	TCP Port Service Multiplexer
2	TCP	UDP	Management Utility
3	TCP	UDP	Compression Process
4	TCP	UDP	Unassigned
5	TCP	UDP	Remote Job Entry
6	TCP	UDP	Unassigned
7	TCP	UDP	Echo
8	TCP	UDP	Unassigned
9	TCP	UDP	Discard
11	TCP	UDP	Active Users
13	TCP	UDP	DAYTIME - (RFC 867)
17	TCP	UDP	Quote of the Day
18	TCP	UDP	Message Send Protocol
19	TCP	UDP	Character Generator
20	TCP		FTP - data
21	TCP		FTP - control (command)
22	TCP	UDP	Secure Shell (SSH)—used for secure logins, file transfers (scp, sftp) and port forwarding
23	TCP		Telnet protocol—unencrypted text communications
25	TCP		Simple Mail Transfer Protocol (SMTP)—used for e-mail routing between mail servers
26	TCP		Unknown Found while scanning website with Nmap. Looks to be SMTP related
34	TCP	UDP	Remote File (RF)—used to transfer files between machines
35	TCP	UDP	Any private printer server protocol
35	TCP	UDP	QMS Magicolor 2 printer server protocol
37	TCP	UDP	TIME protocol
39	TCP	UDP	Resource Location Protocol (RLP)—used for determining the location of higher level services from hosts on a network

41	TCP	UDP	Graphics
42	TCP	UDP	nameserver, ARPA Host Name Server Protocol
42	TCP	UDP	WINS
43	TCP		WHOIS protocol
47	TCP		GRE protocol
49	TCP	UDP	TACACS Login Host protocol
50	TCP	UDP	Encapsulating Security Payload (ESP)
51	TCP	UDP	Authentication Header (AH)
52	TCP	UDP	XNS (Xerox Network Systems) Time Protocol
53	TCP	UDP	Domain Name System (DNS)
54	TCP	UDP	XNS (Xerox Network Systems) Clearinghouse
55	TCP	UDP	ISI Graphics Language (ISI-GL)
56	TCP	UDP	XNS (Xerox Network Systems) Authentication
56	TCP	UDP	Route Access Protocol (RAP)
57	TCP		Mail Transfer Protocol (MTP)
58	TCP	UDP	XNS (Xerox Network Systems) Mail
67		UDP	Bootstrap Protocol (BOOTP) Server; also used by Dynamic Host Configuration Protocol (DHCP)
68		UDP	Bootstrap Protocol (BOOTP) Client; also used by Dynamic Host Configuration Protocol (DHCP)
69		UDP	Trivial File Transfer Protocol (TFTP)
70	TCP		Gopher protocol
79	TCP		Finger protocol
80	TCP	UDP	Hypertext Transfer Protocol (HTTP)
81	TCP		Torpark—Onion routing
82		UDP	Torpark—Control
83	TCP		MIT ML Device
88	TCP	UDP	Kerberos—authentication system
90	TCP	UDP	dnsix (DoD Network Security for Information Exchange) Securit Attribute Token Map
90	TCP	UDP	Pointcast
99	TCP		WIP Message Protocol
101	TCP		NIC host name
102	TCP		ISO-TSAP (Transport Service Access Point) Class 0 protocol
104	TCP	UDP	ACR/NEMA Digital Imaging and Communications in Medicine
105	TCP	UDP	CCSO Nameserver Protocol (Qi/Ph)
107	TCP		Remote TELNET Service protocol
108	TCP	UDP	SNA Gateway Access Server
109	TCP		Post Office Protocol 2 (POP2)
110	TCP		Post Office Protocol 3 (POP3)
111	TCP	UDP	ONC RPC (SunRPC)
113	TCP		ident—user identification system, used by IRC servers to identify users

113	TCP	UDP	Authentication Service (auth)
115	TCP		Simple File Transfer Protocol (SFTP)
117	TCP		UUCP Path Service
118	TCP	UDP	SQL (Structured Query Language) Services
119	TCP		Network News Transfer Protocol (NNTP)—used for retrieving newsgroup messages
123		UDP	Network Time Protocol (NTP)—used for time synchronization
135	TCP	UDP	DCE endpoint resolution
135	TCP	UDP	Microsoft EPMAP (End Point Mapper), also known as DCE/RPC Locator service, used to remotely manage services including DHCP server, DNS server and WINS. Also used by DCOM
137	TCP	UDP	NetBIOS NetBIOS Name Service
138	TCP	UDP	NetBIOS NetBIOS Datagram Service
139	TCP	UDP	NetBIOS NetBIOS Session Service
143	TCP	UDP	Internet Message Access Protocol (IMAP)—used for retrieving, organizing, and synchronizing e-mail messages
152	TCP	UDP	Background File Transfer Program (BFTP)
153	TCP	UDP	SGMP, Simple Gateway Monitoring Protocol
156	TCP	UDP	SQL Service
158	TCP	UDP	DMSP, Distributed Mail Service Protocol
161		UDP	Simple Network Management Protocol (SNMP)
162	TCP	UDP	Simple Network Management Protocol Trap (SNMPTRAP)
170	TCP		Print-srv, Network PostScript
177	TCP	UDP	X Display Manager Control Protocol (XDMCP)
179	TCP		BGP (Border Gateway Protocol)
194	TCP	UDP	IRC (Internet Relay Chat)
199	TCP	UDP	SMUX, SNMP Unix Multiplexer
201	TCP	UDP	AppleTalk Routing Maintenance
209	TCP	UDP	The Quick Mail Transfer Protocol
210	TCP	UDP	ANSI Z39.50
213	TCP	UDP	Internetwork Packet Exchange (IPX)
218	TCP	UDP	Message posting protocol (MPP)
220	TCP	UDP	Internet Message Access Protocol (IMAP), version 3
256	TCP	UDP	2DEV “2SP” Port
259	TCP	UDP	ESRO, Efficient Short Remote Operations
264	TCP	UDP	BGMP, Border Gateway Multicast Protocol
308	TCP		Novastor Online Backup
311	TCP		Mac OS X Server Admin (officially AppleShare IP Web administration)
318	TCP	UDP	PKIX TSP, Time Stamp Protocol
323	TCP	UDP	IMMP, Internet Message Mapping Protocol
350	TCP	UDP	MATIP-Type A, Mapping of Airline Traffic over Internet Protocol
351	TCP	UDP	MATIP-Type B, Mapping of Airline Traffic over Internet Protocol

366	TCP	UDP	ODMR, On-Demand Mail Relay
369	TCP	UDP	Rpc2portmap
370	TCP	UDP	codauth2 – Coda authentication server
370	TCP	UDP	securecast1 – Outgoing packets to NAI's servers, http://www.nai.com/asp_set/anti_virus/alerts/faq.as
371	TCP	UDP	ClearCase albd
383	TCP	UDP	HP data alarm manager
384	TCP	UDP	A Remote Network Server System
387	TCP	UDP	AURP, AppleTalk Update-based Routing Protocol
389	TCP	UDP	Lightweight Directory Access Protocol (LDAP)
401	TCP	UDP	UPS Uninterruptible Power Supply
402	TCP		Altiris, Altiris Deployment Client
411	TCP		Direct Connect Hub
412	TCP		Direct Connect Client-to-Client
427	TCP	UDP	Service Location Protocol (SLP)
443	TCP		HTTPS (Hypertext Transfer Protocol over SSL/TLS)
444	TCP	UDP	SNPP, Simple Network Paging Protocol (RFC 1568)
445	TCP		Microsoft-DS Active Directory, Windows shares
445	TCP		Microsoft-DS SMB file sharing
464	TCP	UDP	Kerberos Change/Set password
465	TCP		Cisco protocol
465	TCP		SMTP over SSL
475	TCP		tcpnethasprv (Aladdin Knowledge Systems Hasp services, TCP/IP version)
497	TCP		Dantz Retrospect
500	TCP		Internet Security Association and Key Management Protocol (ISAKMP)
501	TCP		STMF, Simple Transportation Management Framework – DOT NTCIP 1101
502	TCP	UDP	Modbus, Protocol
504	TCP	UDP	Citadel – multiservice protocol for dedicated clients for the Citadel groupware system
510	TCP		First Class Protocol
512	TCP		Rexec, Remote Process Execution
512		UDP	comsat, together with biff
513	TCP		rlogin
513		UDP	Who
514	TCP		Shell—used to execute non-interactive commands on a remote system (Remote Shell, rsh, remsh)
514		UDP	Syslog—used for system logging
515	TCP		Line Printer Daemon—print service
517		UDP	Talk
518		UDP	NTalk
520	TCP		efs, extended file name server

520		UDP	Routing Information Protocol (RIP)
524	TCP	UDP	NetWare Core Protocol (NCP) is used for a variety of things such as access to primary NetWare server resources, Time Synchronization, etc.
525		UDP	Timed, Timeserver
530	TCP	UDP	RPC
531	TCP	UDP	AOL Instant Messenger, IRC
532	TCP		netnews
533		UDP	netwall, For Emergency Broadcasts
540	TCP		UUCP (Unix-to-Unix Copy Protocol)
542	TCP	UDP	commerce (Commerce Applications)
543	TCP		klogin, Kerberos login
544	TCP		kshell, Kerberos Remote shell
545	TCP		OS/Soft PI (VMS), OS/Soft PI Server Client Access
546	TCP	UDP	DHCPv6 client
547	TCP	UDP	DHCPv6 server
548	TCP		Apple Filing Protocol (AFP) over TCP
550		UDP	new-rwho, new-who
554	TCP	UDP	Real Time Streaming Protocol (RTSP)
556	TCP		Remotefs, RFS, rfs_server
560		UDP	rmonitor, Remote Monitor
561		UDP	monitor
563	TCP	UDP	NNTP protocol over TLS/SSL (NNTPS)
587	TCP		e-mail message submission (SMTP)
591	TCP		FileMaker 6.0 (and later) Web Sharing (HTTP Alternate, also see port 80)
593	TCP	UDP	HTTP RPC Ep Map, Remote procedure call over Hypertext Transfer Protocol, often used by Distributed Component Object Model services and Microsoft Exchange Server
604	TCP		TUNNEL profile, a protocol for BEEP peers to form an application layer tunnel
623		UDP	ASF Remote Management and Control Protocol (ASF-RMCP)
631	TCP	UDP	Internet Printing Protocol (IPP)
636	TCP	UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS)
639	TCP	UDP	MSDP, Multicast Source Discovery Protocol
641	TCP	UDP	SupportSoft Nexus Remote Command (control/listening): A proxy gateway connecting remote control traffic
646	TCP	UDP	LDP, Label Distribution Protocol, a routing protocol used in MPLS networks
647	TCP		DHCP Failover protocol
648	TCP		RRP (Registry Registrar Protocol)
652	TCP		DTCP, Dynamic Tunnel Configuration Protocol
653	TCP	UDP	SupportSoft Nexus Remote Command (data): A proxy gateway connecting remote control traffic
654	TCP		Media Management System (MMS) Media Management Protocol (MMP)

657	TCP	UDP	IBM RMC (Remote monitoring and Control) protocol, used by System p5 AIX Integrated Virtualization Manager (IVM) and Hardware Management Console to connect managed logical partitions (LPAR) to enable dynamic partition reconfiguration
660	TCP		Mac OS X Server administration
665	TCP		sun-dr, Remote Dynamic Reconfiguration
666		UDP	Doom, first online first-person shooter
674	TCP		ACAP (Application Configuration Access Protocol)
691	TCP		MS Exchange Routing
692	TCP		Hyperwave-ISP
694	TCP	UDP	Linux-HA High availability Heartbeat
695	TCP		IEEE-MMS-SSL (IEEE Media Management System over SSL)
698		UDP	OLSR (Optimized Link State Routing)
699	TCP		Access Network
700	TCP		EPP (Extensible Provisioning Protocol), a protocol for communication between domain name registries and registrars (RFC 5734)
701	TCP		LMP (Link Management Protocol (Internet)), a protocol that runs between a pair of nodes and is used to manage traffic engineering (TE) links
702	TCP		IRIS (Internet Registry Information Service) over BEEP (Blocks Extensible Exchange Protocol) (RFC 3983)
706	TCP		Secure Internet Live Conferencing (SILC)
711	TCP		Cisco Tag Distribution Protocol—being replaced by the MPLS Label Distribution Protocol
712	TCP		Topology Broadcast based on Reverse-Path Forwarding routing protocol (TBRPF) (RFC 3684)
712		UDP	Promise RAID Controller
720	TCP		SMQP, Simple Message Queue Protocol
749	TCP	UDP	Kerberos (protocol) administration
750	TCP		rfile
750		UDP	loadav
750		UDP	kerberos-iv, Kerberos version IV
751	TCP	UDP	pump
751	TCP	UDP	kerberos_master, Kerberos authentication
752	TCP		qrh
752		UDP	qrh
752		UDP	passwd_server, Kerberos Password (kpasswd) server
753	TCP		Reverse Routing Header (rrh)
753		UDP	Reverse Routing Header (rrh)
753		UDP	userreg_server, Kerberos userreg server
754	TCP		tell send
754	TCP		krb5_prop, Kerberos v5 slave propagation
754		UDP	tell send

760	TCP	UDP	ns
760	TCP	UDP	krbupdate [kreg], Kerberos registration
782	TCP		Conserver serial-console management server
783	TCP		SpamAssassin spamd daemon
829	TCP		CMP (Certificate Management Protocol)
843	TCP		Adobe Flash socket policy server
860	TCP		iSCSI (RFC 3720)
873	TCP		rsync file synchronisation protocol
888	TCP		cddbp, CD DataBase (CDDb) protocol (CDDBP)—unassigned but widespread use
901	TCP		Samba Web Administration Tool (SWAT)
901	TCP	UDP	VMware Virtual Infrastructure Client (UDP from server being managed to management console)
902	TCP		ideafarm-door 902/tcp self documenting Door: send 0x00 for info
902	TCP		VMware Server Console (TCP from management console to server being Managed)
902		UDP	ideafarm-door
902		UDP	VMware Server Console (UDP from server being managed to management console)
903	TCP		VMware Remote Console
904	TCP		VMware Server Alternate (if 902 is in use, i.e. SUSE linux)
911	TCP		Network Console on Acid (NCA)—local tty redirection over OpenSSH
953	TCP	UDP	Domain Name System (DNS) RNDC Service
981	TCP		SofaWare Technologies Remote HTTPS management for firewall devices running embedded Check PointFireWall-1 software
989	TCP	UDP	FTPS Protocol (data): FTP over TLS/SSL
990	TCP	UDP	FTPS Protocol (control): FTP over TLS/SSL
991	TCP	UDP	NAS (Netnews Administration System)
992	TCP	UDP	TELNET protocol over TLS/SSL
993	TCP		Internet Message Access Protocol over SSL (IMAPS)
995	TCP		Post Office Protocol 3 over TLS/SSL (POP3S)
999	TCP		ScimoreDB Database System
1001	TCP		JtoMB
1002	TCP		Opsware agent (aka cogbot)
1023	TCP	UDP	Reserved



FAST TRACK- JUNE 2010



FAST TRACK- JUNE 2010



FAST TRACK- JUNE 2010



Handwriting practice lines consisting of 20 horizontal rows. Each row is defined by a solid top line, a dashed midline, and a solid bottom line, providing a guide for letter height and placement.

FAST TRACK- JUNE 2010